Programmeringspärmen

Innehåll

Träff 1
Mål med träff 1 , Körexempel Träff 1

in- och utmatning

olika typer av variabler och

matematiska beräkningar

Ex 1-5

```
1   print ("Välkommen till Python!")
2   print("Välkommen")
3   print("till")
4   print("Python!")
5   s1 = "Välkommen"
6   s2 = "till"
7   s3 = "Python!"
8
9   print(s1 + s2 + s3)
10  print()
11  print(s1 + " " + s2 + " " + s3)
12
13  a = 5
14  b = 3
15  print(a + b)
16  print(a - b)
17  print(a * b)
18  print(a / b)
19  print(a % b)
20  print(a // b)
21  tal1 = int(input("Mata in första talet: "))
22  tal2 = int(input("Mata in andra talet: "))
23  print("Summan av tal1 och tal2 är " + str(tal1 + tal2))
24
25  tal1 = input("Mata in första talet: ")
26  tal2 = input("Mata in andra talet: ")
27  print("Summan av tal1 och tal2 är " + str(tal1 + tal2))
```

```
Välkommen till Python!
Välkommen
till
Python!
VälkommentillPython!

Välkommen till Python!
8
2
15
1.6666666666666667
2
1
Mata in första talet: 4
Mata in andra talet: 5
Summan av tal1 och tal2 är 9
Mata in första talet: 4
Mata in andra talet: 5
Summan av tal1 och tal2 är 45
```

Kvadratroten och roten ur ett tal [from math import *]

Ex 6

```
1   from math import *
2   print(sqrt(100))
3   print(pow(2, 3))
```

```
10.0
8.0
```

Uppgift 2

ax + b =c

Indata: a? 5 , b? -3 c? 7 Utdata: x = 2

```
1   print("Programmet löser en ekvation av typen ax + b = c där
    a, b och c är heltal")
2
3   #int före input sätter typen på indata till ett heltal
    (integer)
4   #Om int utelämnas betraktar Python a som en sträng (text) och
5   #kan då inte utföra beräkningen av x.
6   a = int(input("Mata in a: "))
7   b = int(input("Mata in b: "))
8   c = int(input("Mata in c: "))
9
10  x = (c - b) / a
11
12  print("x = ", x)
```

```
Programmet löser en ekvation av typen ax + b = c där a, b och c
 är heltal
Mata in a: 2
Mata in b: 3
Mata in c: 4
x =  0.5
```

Uppgift 3

Summa, differens, produkt och kvot

Indata: Tal 1: 2.5 Tal 2: 7.5 Utdata: Summa: 10.0. Differens: -5.0 Produkt: 18.75 Kvot: 0.3333333333333333

```
1   #float före input tvingar tal1 att betraktas som ett
    decimaltal (även heltal
2   #går bra. float - flyttal) att mata in. Vid inmatning
    används decimalpunkt.
3
4   tal1 = float(input("Mata in första talet: "))
5   tal2 = float(input("Mata in andra talet: "))
6
7   print("Summa: ", (tal1 + tal2))
8   print("Differens: ", (tal1 - tal2))
9   print("Produkt: ", (tal1 * tal2))
10  print("Kvot: ", (tal1 / tal2))
```

```
Mata in första talet: 4.5
Mata in andra talet: 6.5
Summa:  11.0
Differens:  -2.0
Produkt:  29.25
Kvot:  0.6923076923076923
```

Uppgift 4 Indata: Tal? 2.5 Utdata: Tal i kvadrat: 6.25

Olika sätt att kvadrera ett tal

```
1   tal = float(input("Mata in tal som ska kvadreras: "))
2
3   #Olika sätt att kvadrera ett tal.
4   #Rekommendationen är att använda funktionen pow(a, b), (a to
    the power of b),
5   #eftersom den även klarar t ex pow(1/7, 1/3)
6   print("Kvadraten på talet är: ", (tal * tal))
7
8   print("Kvadraten på talet är: ", (pow(tal, 2)))
9
10  print("Kvadraten på talet är: ", (tal ** 2))
```

```
Mata in tal som ska kvadreras: 3
Kvadraten på talet är:  9.0
Kvadraten på talet är:  9.0
Kvadraten på talet är:  9.0
```

Uppgift 5 Indata: Mata in längd på katet 1: 7.2 Mata in längd på katet 2: 9.3 Utdata: Hypotenusans längd: 11.761377470347595

Katet, katet ger hypotenusa

```
1   from math import *
2
3   print("******* PYTHAGORAS SATS *******")
4   print("-------------------------------")
5
6   katet1 = float(input("Mata in längden på katet 1: "))
7   katet2 = float(input("Mata in längden på katet 2: "))
8
9   hypotenusa = sqrt(pow(katet1, 2) + pow(katet2, 2))
10
11  print("Hypotenusans längd:", (hypotenusa))
12
13  #Det går även att skriva beräkningen direkt i printsatsen
    men läsbarheten
14  #ökar om resultatet läggs i en separat variabel. Tänk på hur
    variabelnamnen
15  #väljs. katet1, katet2 ger ökad läsbarhet jämfört med x, y.
16
17  #Detta går också
18
19  print("Hypotenusans längd:", (sqrt(pow(katet1, 2) + pow
    (katet2, 2))))
```

```
******* PYTHAGORAS SATS *******
-------------------------------
Mata in längden på katet 1: 3
Mata in längden på katet 2: 6
Hypotenusans längd: 6.708203932499369
Hypotenusans längd: 6.708203932499369
```

Uppgift 6

Rektangel

Indata: Mata in längden på sida 1: 4
Mata in längden på sida 2: 8.5
Utdata: Rektangelns omkrets: 25.0
Rektangelns area: 34.0

```
1   print("*******     REKTANGEL     *******")
2   print("---------------------------------")
3
4   sida1 = float(input("Mata in längden på sida 1: "))
5   sida2 = float(input("Mata in längden på sida 2: "))
6
7   omkrets = 2 * sida1 + 2 * sida2
8   area = sida1 * sida2
9
10  print("Rektangelsn omkrets:", (omkrets))
11  print("Rektangelsn area:", (area))
```

```
*******     REKTANGEL     *******
---------------------------------
Mata in längden på sida 1: 6
Mata in längden på sida 2: 3
Rektangelsn omkrets: 18.0
Rektangelsn area: 18.0
```

Uppgift 7

Namn och adress

Indata: Förnamn: Filip , Efternamn: Blad Adress: Föreningsgatan 41 Postnummer: 682 01 Ort: Filipstad, Utdata: Filip Blad, Föreningsgatan 41 682 01 Filipstad

```
1   #Inläsning av strängar. Observera att sträng är default för
    input.
2   s1 = input("Skriv in förnamn: ")
3   s2 = input("Skriv in efternamn: ")
4   s3 = input("Skriv in adress: ")
5   s4 = input("Skriv in postnummer: ")
6   s5 = input("Skriv in ort: ")
7
8   #Slå ihop strängar med +
9   print(s1 + " " + s2)
10  print(s3)
11  print(s4 + " " + s5)
```

```
Skriv in förnamn: Anna
Skriv in efternamn: Malmberg
Skriv in adress: Välinge 220
Skriv in postnummer: 655 95
Skriv in ort: Väse
Anna Malmberg
Välinge 220
655 95 Väse
```

Uppgift 8

Timmar, minuter och sekunder

Indata: Antal sekunder? 7398 Utdata: tt:mm:ss: 2:3:18

```
1   print("***** TIDSOMVANDLING *****")
2
3   sek = int(input("Antal sekunder: "))
4   tim = sek // 3600
5   sek = sek % 3600
6   min = sek // 60
7   sek = sek % 60
8
9   print(str(tim)+":"+str(min)+":"+str(sek))
10
11  '''
12  Man kan formulera utskriften så här:
13
14  print("%02i:%02i:%02i" %(tim,min,sek))
15
16  Det betyder %02i betyder att man i strängen reserverar plats
    för en variabel.
17  Variablerna anges sedan i den ordning de ska finnas i
    strängen i en parentes som
18  har ett %-tecken framför sig.
19  '''
```

```
***** TIDSOMVANDLING *****
Antal sekunder: 9748
2:42:28
```

Träff 2

Mål med träff 2

Jämförelseoperatorer och logiska operatorer för att åstadkomma selektion.

Avrunda tal

selektion med if, else och elif

Jämföra två tal

```
"""Selektion med if"""
a = 5
b = 10
if a == b:
  print("a är lika med b")
  #inget händer
"""  Selektion med if- else """

a = 5
b = 10
if a == b:
  print("a är lika med b")
else:
  print("a är skilt från b")

# Selektion med if - elif - elif - … - else
a = int(input("Mata in första talet: "))
b = int(input("Mata in andra talet: "))

if a == b:
  print("a är lika med b")
elif a > b:
  print("a är större än b")
else:
    print("b är större än a")
```

```
a är skilt från b
Mata in första talet: 3
Mata in andra talet: 4
b är större än a
```

```python
#ex 4
"""Logiska operatorer"""

a = int(input("Mata in första talet: "))
b = int(input("Mata in andra talet: "))

if a > 0 and b > 0:
    print("Båda talen är positiva.")

if a > 0 or b > 0:
    print("Minst ett tal är positivt.")

if not(a > 0) and not(b > 0):
    print("Inget av talen är positivt.")

#ex 5
print(1/6)

print(round(1/6,2))

print(round(1/6))
```

```
Mata in första talet: 4
Mata in andra talet: 6
Båda talen är positiva.
Minst ett tal är positivt.
0.16666666666666666
0.17
0
```

Körexempel Träff 2

Uppgift 1 Indata: a? 5

b? -3 Utdata: a är skilt från b

```python
a = int(input("Mata in första talet: "))
b = int(input("Mata in andra talet: "))
if a == b:
    print("a är lika med b")
else:
    print("a är skilt från b")

a = int(input("a? "))
b = int(input("b? "))
#OBS
#Test av likhet görs med ==
if a == b:
    print("a är lika med b")
else:
    print("a är skilt från b")

```

```
Mata in första talet: 7
Mata in andra talet: 8
a är skilt från b
a? 3
b? 4
a är skilt från b
```

Uppgift 2a Indata: a? -3

Vilket tal är störst?

b? 5 Utdata: Minsta talet är -3

```
"""Mata in två tal och skriv ut det största talet"""

x = int(input("Mata in första talet : "))
y = int(input("Mata in andra talet: "))

if x < y:
    print("Talet " + str(y) + " är störst")
else:
    print("Talet " + str(x) + " är störst")


a = int(input("a? "))
b = int(input("b? "))

if a > b:
    print(a)
else:
    #Skriver ut b om b >= a
    print(b)
```

```
Mata in första talet : 7
Mata in andra talet: 9
Talet 9 är störst
a? 5
b? 4
5
```

Uppgift 2b Indata: a? 3 b? 7 c? 3 Utdata: Minsta talet är 3

```
print("Ett program som tar in tre tal och skriver ut det minsta")

x = int(input("x = "))
y = int(input("y = "))
z = int(input("z = "))

if x < y:
    if x < z:
        print("Minsta talet är " + str(x))
    else:
        print("Minsta talet är " + str(z))
else:
    if y < z:
        print("Minsta talet är " + str(y))
    else:
        print("Minsta talet är " + str(z))

a = int(input("a? "))
b = int(input("b? "))
c = int(input("c? "));

#Om a är mindre än b och a är mindre än c -> a är minst
if a < b and a < c:
    print(a)
#Om vi är här vet vi att a inte är minst
elif b < c:
    print(b)
#Kom vi hit kan heller inte b vara minst
else:
    print(c)
```

```
Ett program som tar in tre tal och skriver ut det minsta
x = 5
y = 7
z = 2
Minsta talet är 2
a? 6
b? 9
c? 3
3
```

Uppgift 3a)

Varans vikt och pris

Indata: Antal hekto? 13.5 Utdata: Pris: 120.15 kr

```
print("Ett program som frågar efter antalet hg och därefter\n
skriver ut priset för lösgodis\n")

#Läs in som float. Antal hekto måste inte vara heltal
vikt = float(input("Ange antal hg:"))

#Antal hekto kan inte vara mindre än eller lika 0
if vikt <= 0:
    print("Otillåtet värde")
#Annars om vikt är mindre än 10
elif vikt < 10:
    print("Priset är 9,90 kr/hg")
#Om inget av ovan var sant och vikt är mindre än lika 20
elif vikt <= 20:
    print("Priset är 8,90 kr/hg")
#Om inget av ovan var sant återstår endast
else:
    print("Priset är 7,90 kr/hg")
```

```
https://uppgift-3a-t2.annamalmberg2.repl.run

Ett program som frågar efter antalet hg och därefter
 skriver ut priset för lösgodis

Ange antal hg:7
Priset är 9,90 kr/hg
```

Uppgift 3b) Indata: Antal kronor? 153 Utdata: Antal hekto: 17.19

```
antalKronor = float(input("Antal kronor: "))
pris = 0
#Kunden måste ha med sig mer än 158 kr för att kunna köpa för 7.90
kr/hg
if antalKronor > 158:
    pris = 7.9
#Om kunden har mindre än 89 blir priset alltid 9.90/hg
elif antalKronor < 89:
    pris = 9.9
#I alla andra fall kostar godiset 8.90/hg
else:
    pris = 8.9

print("Antal hekto: " + str(antalKronor / pris))
```

```
https://uppgift-3b-t2.annamalmberg2.repl.run

Antal kronor: 120
Antal hekto: 13.48314606741573
```

Uppgift 4

E-poäng, c-poäng, a-poäng poäng ger provbetyg

Indata: E-poäng? 27 C-poäng? 10 A-poäng? 9 Utdata: Provbetyg: D

```
1   e = int(input("E-poäng? "))
2   c = int(input("C-poäng? "))
3   a = int(input("A-poäng? "))
4
5   sum = e + c + a
6
7   #Börja med betyget A!
8   if sum >= 64 and a >= 12:
9       print("Provbetyg: A")
10  elif sum >= 54 and a >= 7:
11      print("Provbetyg: B")
12  elif sum >= 44 and sum < 44 and (c + a) >= 20:
13      print("Provbetyg: C")
14  elif sum >= 32 and (c + a) >= 11:
15      print("Provbetyg: D")
16  elif sum >= 20:
17      print("Provbetyg: E")
18  #Om inget villkor av de ovan var uppfyllt finns endast
19  #en möjlighet kvar.
20  else:
21      print("Provbetyg: F")
```

```
https://uppgift-4-t2.annamalmberg2.repl.run

E-poäng? 56
C-poäng? 23
A-poäng? 30
Provbetyg: A
```

Uppgift 5a

Andragradsekvationen

Indata: a? 4 , b? -8 c? -5
Utdata: x1 = -0.5 x2 = 2.5

```
1   import math
2
3   print("Programmet löser en andragradsekvation av typen ax\u00b2+bx
    +c=0\n")
4   a = float(input("a = "))
5   b = float(input("b = "))
6   c = float(input("c = "))
7
8   #Division med a
9   b = b / a
10  c = c / a
11
12  # Vi löser ekvationen med p-q formeln
13  d = (b / 2)**2 - c   #beräknar värdet för uttrycket under
    kvadratroten ...
14  x = - b / 2      #symmetrilinjen
15
16  if d < 0:
17      print("Reella lösningar saknas")
18
19  elif  d == 0:
20      print("En dubbelrot: x =" , round(x,2))
21  else:
22      x1 = round(x+math.sqrt(d),2)
23      x2 = round(x-math.sqrt(d),2)
24      print("Två reella lösningar: x\u2081 =", x1 ," och     x\u2082
        =", x2)
```

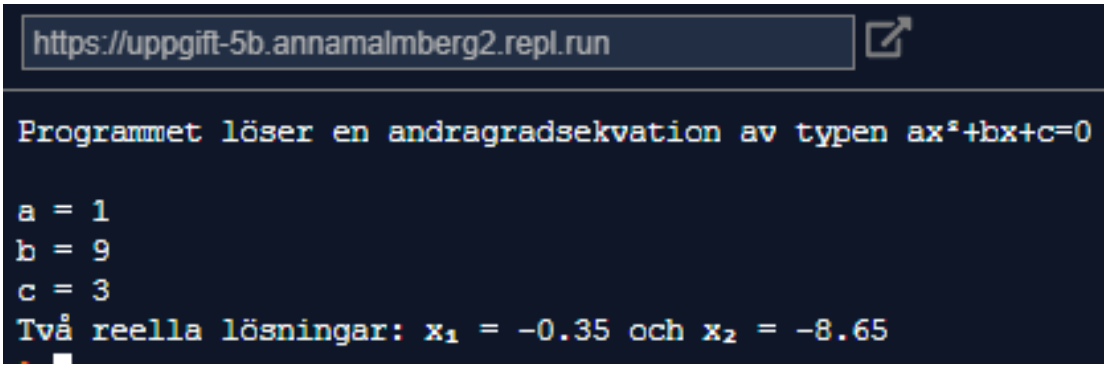
```
https://uppgift-5a-t2.annamalmberg2.repl.run

Programmet löser en andragradsekvation av typen ax²+bx+c=0

a = 1
b = 6
c = 1
Två reella lösningar: x₁ = -0.17  och     x₂ = -5.83
```

Uppgift 5b Indata: Samma indata som för 5a ger samma utdata enligt ovan.
Indata: a? 3 b? -12 c? 15 Utdata: x1 = 2-i x2 = 2+i

```
1   from math import *
2
3   print("Programmet löser en andragradsekvation av typen ax\u00b2+bx
    +c=0\n")
4   a = float(input("a = "))
5   b = float(input("b = "))
6   c = float(input("c = "))
7
8   #Division med a
9   b = b / a
10  c = c / a
11
12  # Vi löser ekvationen med p-q formeln
13  d = (b / 2)**2 - c   #beräkna värdet för uttrycket under
    kvadratroten ...
14  x = - b / 2          # symmetrilinjen
15
16  if d < 0:  # Komplexa lösningar
17      i = round(sqrt(-d),2)  # beräknar imaginära delen
18      print("Två komplexa lösningar: x\u2081 = " + str(round(x,2)) + "
        + " + str(i) + "i och x\u2082 = " + str(round(x,2)) + " - " +str
        (i) + "i.")
19
20  elif  d == 0:
21      print("En dubbelrot: x = " + str(round(x,2)))
22  else:
23      x1 = round(x+sqrt(d),2)
24      x2 = round(x-sqrt(d),2)
25      print("Två reella lösningar: x\u2081 = " + str(x1) + " och
        x\u2082 = " + str(x2))
```



Träff 3

Nedan finner du presentationen med genomgång, exempel och uppgifter. Du finner också körexempel till uppgifterna så att du kan testa dina lösningar. Slutligen finns även lösningsförslag.

Mål med träff 3

For-loopar för upprepning

Slumpa fram heltal

Python.Undervisningstillfälle 3. Loopar
När något ska upprepas ett begränsat antal gånger är det smidigt med en loop.
Den första typen av loop vi tittar på är en for-loop.
for i in range(start, slut + 1, steglängd):

instruktion 1

.

.

.

Exempel 1

Listor på olika sätt

Undersök vad som händer när du kör koden. Vilka likheter skillnader finns?

a) for i in range(0,10,1):

print(i)

b) for i in range(0,10,3):

print(i)

c) for i in range(10,0,-1):

print(i)

d) for i in range(10):

print(i)

```
#1a
for i in range(0,10,1):
    print(i)

#1b
for i in range(0,10,3):
    print(i)
#1c
for i in range(10,0,-1):
    print(i)
#1d
for i in range(10):
    print(i)
```

```
1
2
3
4
5
6
7
8
9
0
3
6
9
10
9
8
7
6
5
4
3
2
1
0
1
2
3
4
5
6
7
8
9
```

Exempel 2

Aritmetisk summa

Programmet nedan beräknar summan av de 1000 första heltalen. Hur fungerar koden?

```
summa = 0
for k in range(1,1001,1):
summa = summa + k
print(summa)
```

Vad händer med summan vid instruktionen summa = summa + k?

```
1   summa = 0
2   for k in range(1,1001,1):
3       summa = summa + k
4   print(summa)
```

```
500500
```

Uppgift 1

Listor med for loop

a) Skriv ett program som skriver ut alla jämna heltal mellan 2 och 50.

```
1   #for-loopen startar på 2, slutar på 50 i steg om 2
2   for i in range(2,51,2):
3       print(i)
```

```
2
4
6
8
10
12
14
16
18
20
22
24
26
28
30
32
34
36
38
40
42
44
46
48
50
```

b) Skriv ett program som skriver ut alla jämna heltal mellan 1 och 20 baklänges.

```
1   #for-loopen startar på 20, slutvärde (0 eller 1 funkar) och steg -2
2   for i in range(20, 0, -2):
3       print(i)
```

```
20
18
16
14
12
10
8
6
4
2
```

c) Skriv ett program som beräknar “n!”.

n! skriva ut fakulteten av ett tal

```
1   #Vi bygger upp en produkt och startar då på 1
2   #Programmet klarar då 0! = 1! = 1 utan att gå in i loopen
3   fakultet = 1
4
5   n = int(input("Ange det tal du vill beräkna fakulteten för (>=0): "))
6   #Bilda produkten av alla tal från 1 till n
7   #Om inget steg anges är steg alltid 1
8   for i in range(1, n + 1):
9       fakultet = fakultet * i
10  print(str(n) + "! = " + str(fakultet))
```

```
Ange det tal du vill beräkna fakulteten för (>=0): 6
6! = 720
```

d) Skriv ett program som beräknar den aritmetiska summan med startvärde a, steglängd b och n st termer.

Aritmetisk summa med startvärde steglängd och antal termer

```
1   a = int(input("Ange startvärde: "))
2   b = int(input("Ange steglängd: "))
3   n = int(input("Ange antalet termer: "))
4
5   #En summa ska byggas -> starta på noll
6   sum = 0
7
8   #I lösningsförslaget beräknas den sista termen vilket inte är
    nödvändigt
9   #(se alternativt lösningsförslag nedan)
10  for i in range(a, a + (n - 1) * b + 1, b):
11      sum = sum + i
12
13  print("Summa:", sum)
14
15  #Låt i hålla antalet termer
16  sum = 0
17  for i in range(n):
18      #Öka sum med nästkommande tal i talföljden
19      sum = sum +  a + i * b
20  print("Summa: ", sum)
```

```
Ange startvärde: 1
Ange steglängd: 1
Ange antalet termer: 100
Summa: 5050
Summa:  5050
```

e) Skriv ett program som frågar efter ett heltal n > 0 och därefter beräknar summan

Aritmetisk summa av kvoter med n i nämnaren och skilda +/-

1 - 1/2 + 1/3 - 1/4 + … + 1/n. (Överkurs)

```
1   n = int(input("Ange ett heltal: "))
2   sum = 0
3   faktor = 1
4   for i in range(1, n + 1):
5       sum = sum + faktor / i
6       #Byt tecken på faktor eftersom termerna växlar tecken
7       faktor = -faktor
8
9   #Summan konvergerar (mycket långsamt...) mot ln2
10  print(sum)
11
12  #Alternativ kod
13  #n = int(input("Ange ett heltal: "))
14  #sum = 0
15  #for i in range(1, n + 1):
16  #    if i % 2 == 0:
17  #        sum = sum - 1 / i
18  #    else:
19  #        sum = sum + 1 / i
20  #print(sum)
```

```
Ange ett heltal: 6
0.6166666666666666
```

Exempel 3 - nästlade loopar

Nästlade lopar eller två olika listor

Vi kan kombinera två loopar för att få en dubbel uppräkning. Testa programmet nedan och se vad det gör.

for i in range(5):, for j in range(5):, print(i, j, end = “ ”), print()

Instruktionen end = “ ” gör så att utskriften fortsätter på samma rad fast med 3 mellanslag mellan utskrifterna.

```
1   for i in range(5):
2       for j in range(5):
3           print(i, j, end = "   ")
4       print()
```

```
0 0   0 1   0 2   0 3   0 4
1 0   1 1   1 2   1 3   1 4
2 0   2 1   2 2   2 3   2 4
3 0   3 1   3 2   3 3   3 4
4 0   4 1   4 2   4 3   4 4
```

Exempel 4 – slumptal

Slumpa fram tal

Genom att använda import random får vi tillgång till slumpgeneratorer. Vi visar här hur du kan slumpa fram ett heltal.

import random, n = random.randint(1, 6) #Slumpar fram ett heltal 1 till 6., print(n)

Det som skrivs bakom tecknet # räknas som en kommentar och kommer inte köras.

```
1   import random
2
3   n = random.randint(1,6) #Slumpar fram ett heltal 1 till 6.
4   print(n)
```

```
4
```

Exempel 5

Sannolikhet att få en 6:a

För att beräkna sannolikheten för att få en sexa “experimentellt” så kan man använda följande program. Vad händer vid varje steg i programmet?

import random, antalSexor = 0, antalSlag = int(input(“Ange antal slag: ”)), for i in range(antalSlag):, if random.randint(1,6) == 6:, antalSexor = antalSexor + 1, print(antalSexor/antalSlag * 100,“% sexor.”), Kör programmet några gånger. Vad kan du observera med resultatet?

```
1   import random
2   antalSexor = 0
3   antalSlag = int(input("Ange antal slag: "))
4   for i in range(antalSlag):
5       if random.randint(1,6) == 6:
6           antalSexor = antalSexor + 1
7   print(round(antalSexor/antalSlag * 100,2), "% sexor.")
```

```
Ange antal slag: 10000
16.85 % sexor.
```

Uppgifter till nästa träff

2. En modell för lönesättning ger ett årligt påslag på månadslönen med först 3 % och därefter 200 kr. Skriv ett program som frågar efter ingångslön och skriver ut månadslönen år n.

Lön efter n år

```
1   #Ange ingångslönen
2   lon = int(input("Ingångslönen är: "))
3   n = int(input("Ange antalet år: "))
4
5   #Beräkna ny lön
6   for i in range(1, n + 1):
7       lon = lon * 1.03 + 200
8
9   #Skriv ut den nya månadslönen.
10  print("Månadslönen i slutet av år "+ str(n)+ " är: "+str(lon))
```

```
Ingångslönen är: 32000
Ange antalet år: 4
Månadslönen i slutet av år 4 är: 36853.007320000004
>
```

3. Utöka uppgift 2 så att programmet skriver ut nettolönen. Räkna med 30 % skatt för inkomster upp till och med 35 000 kr. På den del av inkomsten som överstiger 35 000 kr betalas 50 % skatt.

nettolön

```
1   #Ange ingångslönen
2   lon = int(input("Ingångslönen (brutto) är: "))
3   n = int(input("Ange antalet år: "))
4
5   #Beräkna ny lön
6   for i in range(1, n + 1):
7       lon = lon * 1.03 + 200
8
9   #Beräknar skatt
10  if lon <= 35000:
11      lon = 0.7 * lon
12  elif lon > 35000:
13      lon = lon - 0.5 * (lon - 35000) - 0.3 * 35000
14
15  #Skriv ut den nya månadslönen.
16  print("Månadslönen (netto) år "+ str(n)+ " är: "+str(lon))
```

```
Ingångslönen (brutto) är: 32000
Ange antalet år: 4
Månadslönen (netto) år 4 är: 25426.503660000002
>
```

4. Skriv ett program som simulerar ett kast med 2 tärningar. Låt simuleringen upprepas 10000 gånger. Räkna hur många gånger tärningarna visar samma antal prickar. Skriv ut sannolikheten för att detta sker.

Sannolikhet för kast med 2 st tärningar

```
1   from random import randint
2
3   raknare = 0
4
5   n = int(input("Ange antalet kast: "))
6
7   #Kör loopen från 0 till n - 1, dvs n gånger
8   for i in range(n):
9       #randint(a, b) slumpar fram ett heltal x, a <= x <= b
10      tarning1 = randint(1, 6)
11      tarning2 = randint(1, 6)
12      #Om tärningarna visar lika
13      if(tarning1 == tarning2):
14          #Räkna antalet gynnsamma utfall
15          raknare = raknare + 1
16
17  #Den simulerade sannolikheten bör såklart konvergera mot 1/6
18  print("Sannolikheten för att båda tärningarna visar samma är: " + str
    (raknare/n * 100) + " %.")
```

```
Ange antalet kast: 6
Sannolikheten för att båda tärningarna visar samma är: 16.666666666666664
%.
```

Uppgifter till nästa träff

5. Skriv ett program som redovisar sidlängderna på alla egyptiska trianglar med sidlängder mindre eller lika med 100. En egyptisk triangel är en rätvinklig triangel där alla sidor är heltal. Använd nästlade for-loopar.

```
kvadratsumma=0
for i in range(1,101):
    #Låt näslade loopen starta på i för att undvika dubletter
    for j in range(i,101):
        #Låt näslade loopen starta på j för att undvika dubletter
        for k in range(j,101):

            #Pythagoras sats
            kvadratsumma = i ** 2 + j ** 2
            if kvadratsumma == k ** 2:
                print("Triangeln med sidorna " + str(i)+ ", " + str
                (j) + " och " +str(k) + " är en egyptisk triangel.")


#Alternativt med två loopar
from math import *
for i in range(1, 101):
    for j in range(i, 101):
        #Om hypotenusan är ett heltal
        hypotenusan = sqrt(pow(i, 2) + pow(j, 2))
        #Skriv ut triangelns sidor
        if hypotenusan == int(hypotenusan) and hypotenusan <= 100:
            print(i, j, int(hypotenusan))
```

```
Triangeln med sidorna 3, 4 och 5 är en egyptisk triangel.
Triangeln med sidorna 5, 12 och 13 är en egyptisk triangel.
Triangeln med sidorna 6, 8 och 10 är en egyptisk triangel.
Triangeln med sidorna 7, 24 och 25 är en egyptisk triangel.
Triangeln med sidorna 8, 15 och 17 är en egyptisk triangel.
Triangeln med sidorna 9, 12 och 15 är en egyptisk triangel.
Triangeln med sidorna 9, 40 och 41 är en egyptisk triangel.
Triangeln med sidorna 10, 24 och 26 är en egyptisk triangel.
Triangeln med sidorna 11, 60 och 61 är en egyptisk triangel.
Triangeln med sidorna 12, 16 och 20 är en egyptisk triangel.
Triangeln med sidorna 12, 35 och 37 är en egyptisk triangel.
Triangeln med sidorna 13, 84 och 85 är en egyptisk triangel.
```
o s v

```
3 4 5
5 12 13
6 8 10
7 24 25
8 15 17
9 12 15
9 40 41
10 24 26
11 60 61
12 16 20
12 35 37
13 84 85
```
o s v

```
48 55 73
48 64 80
51 68 85
54 72 90
57 76 95
60 63 87
60 80 100
65 72 97
```

Träff 4

Nedan finner du presentationen med genomgång, exempel och uppgifter. Du finner också körexempel till uppgifterna så att du kan testa dina lösningar. Slutligen finns även lösningsförslag.

Att kunna använda while-loopar för upprepning

Python. Undervisningstillfälle 4. While-loopar. När man inte vet hur många gånger något ska upprepas så använder man en while-loop istället för en for-loop. En sådan loop kommer att köras så länge villkoret som styr loopen är sann. while villkor: instruktion 1

Exempel 1

Upprepning nr

För att loopen ska sluta måste villkorsvariabelns värde ändras inne i loopen. Villkorsvariabeln måste även definieras innan loopen. Testkör exemplet och försök förutsäga utskriften.
i = 1
while i <= 5:
print(“Upprepning nummer”, i)
i = i + 1 Blev utskriften vad du förväntade dig?

```
1   i = 1
2   while i <= 5:
3       print("Upprepning nummer", i)
4       i = i + 1
```

```
Upprepning nummer 1
Upprepning nummer 2
Upprepning nummer 3
Upprepning nummer 4
Upprepning nummer 5
```

Exempel 2

Beräkna när jag slår in 0

Programmet nedan beräknar medelvärdet av ett antal positiva heltal som matas in av användaren. Inmatningen avbryts när en nolla skrivs in.

antal = 0, summa = 0, tal = int(input(“Mata in ett positivt heltal: ”)), while tal != 0:, antal = antal + 1 #Vad händer med antalet?, summa = summa + tal #Vad händer med summan?, tal = int(input(“Mata in ett positivt heltal: ”)), if antal != 0:, print(“Medelvärdet är”, summa/antal)

```
1   antal = 0
2   summa = 0
3   tal = int(input("Mata in ett positivt heltal: "))
4   while tal != 0:
5       antal = antal + 1      #Vad händer med antalet?
6       summa = summa + tal    #Vad händer med summan?
7       tal = int(input("Mata in ett positivt heltal: "))
8   if antal != 0:
9       print("Medelvärdet är", summa/antal)
```

```
Mata in ett positivt heltal: 6
Mata in ett positivt heltal: 9
Mata in ett positivt heltal: 7
Mata in ett positivt heltal: 0
Medelvärdet är 7.333333333333333
```

Uppgift 1 - Lös följande uppgifter med en while-loop.

Lista med while-loop

a) Skriv ett program som skriver ut alla jämna heltal mellan 2 och 50.

```
1   tal = 2
2
3   #Så länge tal är mindre än eller lika 50
4   while tal <= 50:
5       print(tal)
6       #Minska tal med 2
7       tal = tal + 2
```

```
2
4
6
8
10
12
14
16
18
20
22
24
26
28
30
32
34
36
38
40
42
44
46
48
50
```

b) Skriv ett program som skriver ut alla jämna heltal mellan 1 och 20 baklänges.

```
1   tal = 20
2   #Så länge tal är större än noll
3   while tal > 0:
4       print(tal)
5       #Minska tal med 2
6       tal = tal - 2
```

```
20
18
16
14
12
10
8
6
4
2
```

n! med whileloop

c) Skriv ett program som beräknar “n!”.

```
1   n = int(input("n? "))
2   spara_n = n
3
4   #Vi ska bygga en produkt. Starta därför fakultet med 1
5   fakultet = 1
6
7
8   if n < 0:
9       print("n! är ej definierat för negativa tal.")
10      #0! är lika med 1 per definition
11  elif n == 0:
12      print(str(n) + "! =", fakultet)
13  else:
14      #För fakulteter av
15      while n >= 1:
16          #Låt fakulteten byggas upp genom att multiplicera 1 med alla
            faktorer 1 - n
17          fakultet = fakultet * n
18          n = n - 1
19      print(str(spara_n) + "! =", fakultet)
```

```
n? 7
7! = 5040
```

Aritmetisk summa med while-loop

d) Skriv ett program som beräknar den aritmetiska summan med startvärde a,

steglängd b och n st termer.

```
a = int(input("Ange startvärde: "))
b = int(input("Steglängd: "))
n = int(input("Antal termer: "))

summa = 0
i = a

#Så länge det finns termer att addera
while n > 0:
    summa = summa + i
    #Öka summan med b
    i = i + b
    #Och då har en term lagts till och det är n - 1 kvar
    n = n - 1

#Skriv ut summan
print("Summa:", summa)
```

```
Ange startvärde: 1
Steglängd: 1
Antal termer: 100
Summa: 5050
```

Summera kvoter med olika tecken tills 1/n

e) Skriv ett program som frågar efter ett heltal n > 0 och därefter beräknar

summan 1 - 1/2 + 1/3 - 1/4 + … + 1/n.

```
summa = 0
faktor = 1
i = 1

#Mata in antal termer som ska summeras
n = int(input("Antal termer: "))

#Så länge antal adderade termer (i) är mindre än antal termer som ska summeras (n)
while i <= n:
    #Öka summa med termen faktor / i
    summa = summa + faktor / i

    #faktor byter tecken vid varje loopvarv eftersom termernas tecken växlar
    faktor = -faktor

    #Öka antalet termer med 1
    i = i + 1

#Skriv ut summan
print("Summa: ", summa)
```

```
Antal termer: 100
Summa:  0.688172179310195
```

Euklides algoritm

Exempel 3 - Euklides algoritm. Programmet nedan beräknar största gemensamma delaren till två tal. Hur fungerar programmet?
a = 10, b = 2
t1 = max(a,b) #Funktionerna max()/min() returnerar
t2 = min(a,b) #största/minsta värdet av talen mellan
r = 1 #parenteserna. while r != 0:, r = t1 % t2, t1 = t2, t2 = r
print(“Den största gemensamma delaren till ” + str(a) +“ och ” + str(b) + “ är ” + str(t1))

```
1   a = 10
2   b = 2
3   t1 = max(a,b)     #Funktionerna max()/min() returnerar
4   t2 = min(a,b)     #största/minsta värdet av talen mellan
5   r = 1         #parenteserna.
6   while r != 0:
7       r = t1 % t2
8       t1 = t2
9       t2 = r
10  print("Den största gemensamma delaren mellan " + str(a) + " och " +
    str(b) + " är " + str(t1))
11
```

```
Den största gemensamma delaren mellan 10 och 2 är 2
```

Rotalgoritm med while

Uppgift 2 – Rotalgoritm. En algoritm för att beräkna roten ur ett tal kan formuleras enligt följande: Mata in talet, t Om t < 0 så avbryts programmet (Det går inte att dra roten ur ett negativt tal.).
Sätt a till t, Sätt b till 1, Så länge |a – b| > 0.00000001, a = (a + b) / 2, b = t / a, Skriv ut (a + b) / 2Skriv programmet ovan. Absolutbelopp i Python beräknas med funktionen abs(tal).

```
1   t = float(input("Tal att dra roten ur: "))
2
3   if t < 0:
4       print("Kan ej dra roten ur ", t)
5   #Algoritmen stänger in roten ur t
6   else:
7       a = t
8       b = 1
9       #Beräknar abslolutbeloppet av två på varandra följande
        gissningar av sqrt(t)
10      #Observera att beloppet måste kontrolleras för att hittar roten
        ur tal om
11      #0 <= tal < 1
12      while(abs(a - b) > 0.00000001):
13          a = (a + b) / 2
14          b = t / a
15
16      #Ytterligare en skärpning av den beräknade roten
17      print("Roten ur ", t, " = ", (a + b) / 2)
```

```
Tal att dra roten ur: 10
Roten ur  10.0  =  3.162277660168379
> >
```

Spelet, gissa ett heltal

Spela spelet nedan två och två. Spelare 1 tänker på ett heltal mellan 1 och 100. Spelare 2 gissar vilket heltal spelare 1 tänker på. Spelare 1 ger feedback om det gissade talet är för stort, för litet eller korrekt. Spelare 2 gör ny gissning... Hur många gissningar bör du maximalt behöva?

Exempel 4 - Gissa ett heltal. Nu ska du spela mot datorn. Spela spelet ett antal gånger. Fundera över vad programmet gör. import random, tal = random.randint(1, 1000), gissning = int(input(“Vilket tal gissar du på? “)), antalGissningar = 1, while gissning != tal:, if gissning > tal: print(“Gissningen är för hög.”), else:, print(“Gissningen är för låg.”), gissning = int(input(“Vilket tal gissar du på? “)), antalGissningar = antalGissningar + 1, print(“Du behövde ” + str(antalGissningar) + “ gissningar”)

```
1   import random
2   tal = random.randint(1, 1000)
3   gissning = int(input("Vilket tal gissar du på? "))
4   antalGissningar = 1
5   while gissning != tal:
6     if gissning > tal:
7       print("Gissningen är för hög.")
8     else:
9       print("Gissningen är för låg.")
10    gissning = int(input("Vilket tal gissar du på? "))
11    antalGissningar = antalGissningar + 1
12  print("Du behövde " + str(antalGissningar) + " gissningar.")
```

```
Vilket tal gissar du på? 100
Gissningen är för låg.
Vilket tal gissar du på? 300
Gissningen är för låg.
Vilket tal gissar du på? 500
Gissningen är för hög.
Vilket tal gissar du på? 400
Gissningen är för hög.
Vilket tal gissar du på? 350
Gissningen är för hög.
Vilket tal gissar du på? 320
Gissningen är för låg.
Vilket tal gissar du på? 340
Gissningen är för hög.
Vilket tal gissar du på? 330
Gissningen är för hög.
Vilket tal gissar du på? 325
Gissningen är för hög.
Vilket tal gissar du på? 322
Gissningen är för hög.
Vilket tal gissar du på? 321
Du behövde 11 gissningar.
```

Uppgifter till nästa träff

3. Datorn ska gissa! Skriv ett program som gissar vilket tal, x, 1 <= x <= 1000, som användaren tänker på. Du tänker på ett tal. Datorn börjar med en gissning. Du skriver in följande för att informera programmet om gissningen:

- Gissningen är för liten

+ Gissningen är för stor

0 Gissningen är korrekt

Programmet ska även skriva ut det antal gissningar som programmet behövde.

```
7   #Så länge programmet inte gissar rätt tal
8   while feedback != "0":
9       #Binär sökning
10      #Programmet gissar på medelvärdet av gränserna för att stänga in
11      #talet som användaren tänker på
12      #Observera // eftersom gissning alltid är ett heltal
13      guess = (lowerLimit + upperLimit) // 2
14      print("Jag gissar på", guess)
15
16      #Och då ökar vi antalet gissningar med 1
17      nbrOfGuesses = nbrOfGuesses + 1
18
19      #Tala om för programmet om gissningen var för hög/för låg
20      feedback = input("+/-/0: ")
21      #Om gissningen var för låg
22      if feedback == "-":
23          #så vet vi ju att talet måste vara minst guess + 1
24          lowerLimit = guess + 1
25      #Om gissningen var för hög
26      elif feedback == "+":
27          #så vet vi ju att talet måste vara högst guess + 1
28          upperLimit = guess - 1
29      elif feedback != "0":
30          #Om man slinter på tangentbordet...
31          print("Felaktig inmatning")
32          nbrOfGuesses -= 1
33
34
35  print("Jag behövde ", nbrOfGuesses, " gissningar.")
```

```
Jag gissar på 500
+/-/0: 7
Felaktig inmatning
Jag gissar på 500
+/-/0: 500
Felaktig inmatning
Jag gissar på 500
+/-/0: 450
Felaktig inmatning
Jag gissar på 500
+/-/0: 550
Felaktig inmatning
Jag gissar på 500
+/-/0: 501
Felaktig inmatning
Jag gissar på 500
+/-/0: 499
Felaktig inmatning
Jag gissar på 500
+/-/0: 498
Felaktig inmatning
Jag gissar på 500
+/-/0: 502
Felaktig inmatning
Jag gissar på 500
+/-/0:
```

Uppgifter till nästa träff

Ange delare i ett tal

4. a) Skriv ett program som genom att loopa igenom alla positiva heltal mindre än ett givet tal skriver ut dess delare.

```
1   tal = int(input("Ange tal: "))
2
3   #Undvik den triviala delaren 1
4   for i in range(2, tal):
5       #Om heltalsdivision med i ger resten 0
6       if(tal % i == 0):
7           #så var i en faktor i tal
8           print(i)
```

```
Ange tal: 35
5
7
```

Avgör om primtal

b) Skriv ett program som avgör om ett tal är ett primtal.

```
1   from math import *
2
3   tal = int(input("Ange tal (>=2): "))
4
5   #Det räcker med att kontrollera delare upp till roten ur tal
6   #för att avgöra om tal är delbart
7   högsta_primtalsfaktor = int(sqrt(tal))
8
9   #Antag att tal är ett primtal och visa motsatsen om annat
10  är_primtal = True
11
12  for i in range(2, högsta_primtalsfaktor + 1):
13      #Tänk efter vad som händer
14      #är_primtal förblir True om det redan var True och tal inte är jämnt
        delbart med i
15      är_primtal = är_primtal and tal % i != 0
16
17  #Och skriv ut
18  if är_primtal:
19      print(tal, "är ett primtal")
20  else:
21      print(tal, "är ej ett primtal")
```

```
Ange tal (>=2): 17
17 är ett primtal
```

c) Skriv ett program som hittar och skriver ut de n första primtalen. Talet n ska matas in av användaren.

Skriv ut de n första primtalen

```python
from math import *

n = int(input("Antal primtal: "))
nbrOfPrimes = 0

if n <= 0:
    print("Du vill inte se några primtal")
else:
    #Talet 2 behandlas separat eftersom det är det enda jämna primtalet
    print(2)
    nbrOfPrimes = 1

    #Nästa tal att prova
    number = 3

    while nbrOfPrimes < n:
        #Antag först att number är ett primtal
        isPrime = True
        sqrtOfNumber = int(sqrt(number))
        for i in range(2, sqrtOfNumber + 1):
            #Om number är delbart...
            if number % i == 0:
                #...så är det inget primtal
                isPrime = False
                #break avbryter for-loopen då delbarhet är konstaterad.
                #Sparar bara tid, det funkar lika bra utan break
                break

        if isPrime:
            print(number)
            nbrOfPrimes = nbrOfPrimes + 1
        #Öka number med 2 eftersom alla primtal > 2 är udda
        number = number + 2
```

```
Antal primtal: 9
2
3
5
7
11
13
17
19
23
```

```python
#nästa variant
from math import *

n = int(input("Antal primtal (>0): "))
antal_primtal = 0

#Talet 2 behandlas separat eftersom det är det enda jämna primtalet
tal = 2
if tal == 2:
    antal_primtal = 1
    print(tal)

    #Nästa tal att prova
    tal = 3

    while antal_primtal < n:
        #Antag först att tal är ett primtal
        är_primtal = True
        #Högsta möjliga primtalsfaktor
        roten_ur_tal = int(sqrt(tal))
        for i in range(2, roten_ur_tal + 1):
            #är_primtal förblir True om det redan var True och tal inte är
            jämnt delbart med i
            är_primtal = är_primtal and tal % i != 0
        #Om det var ett primtal
        if är_primtal:
            print(tal)
            antal_primtal = antal_primtal + 1
        #Öka tal med 2 eftersom alla primtal > 2 är udda
        tal = tal + 2
```

```
Antal primtal (>0): 9
2
3
5
7
11
13
17
19
23
```

5. När Askungen var ute och vandrade i skogen träffade hon den goda fen. Fen sade till Askungen:
- Säg vad du önskar dig och jag ska uppfylla det.
- Då önskar jag att alltid förbli vid den ålder jag är vid idag.
När Askungen kom hem och berättade vad som hänt för sina båda äldre styvsystrar blev Begonia, 8 år äldre än Askungen, rasande. Hon utbrast:
- Du vet ju att din styvfar ger oss produkten av våra tre åldrar i daler att dela på varje år.
- Just det, utbrast mellansystern Fuxia. Med ditt självska tilltag kommer vi tillsammans att förlora 1382 daler de två nästföljande åren!

Hur gamla är Askungen och hennes båda styvsystrar?

```
#a - Askungens ålder
a = -1
#Differensen av summan av produkterna då askungen blir äldre jämfört
#med då hon inte blir det
differens = 0

#Så länge korrekt differens inte hittats
while differens != 1382:
    #Öka Askungens ålder med 1
    a = a + 1
    #Begonia är alltid 8 år äldre än Askungen
    b = a + 8

    f = a
    #Så länge ej korrekt differens och fuxia är yngre än Begonia
    while differens != 1382 and f < b:
        #Öka Fuxias ålder med 1
        f = f + 1
        #Beräkna differensen vid de aktuella åldrarna
        differens = (a+1)*(f+1)*(b+1)+(a+2)*(f+2)*(b+2)-a*(f+1)*(b+1)-a*(f+2)*(b
        +2)

print("Askungen är", a, "år")
print("Fuxia är", f, "år")
print("Begonia är", b, "år")
```

```
Askungen är 15 år
Fuxia är 17 år
Begonia är 23 år
```

Träff 5

Nedan finner du presentationen med genomgång, exempel och uppgifter. Du finner också körexempel till uppgifterna så att du kan testa dina lösningar. Slutligen finns även lösningsförslag.

Mål med träff 5

Att kunna använda listor för lagring av data [5]

Att kunna använda grundläggande statistikfunktioner [5]

Python, Undervisningstillfälle 5, Listor, I Python kan man lagra data i en lista.
lista = [element 0, element 1, ...] Elementen i listan är numrerade så att första elementet har index 0.

Exempel 1

Skriva ut listelement [5]

I programmet nedan skapas en lista, sedan skrivs ett antal saker ut. Försök förutse utskriften innan du kör exemplet!

lista = [1, 2, 3, 4]
print(lista[0])
print(lista[1] * lista[3])
print(lista)

```
1   lista = [1, 2, 3, 4]
2   print(lista[0])
3   print(lista[1] * lista[3])
4   print(lista)
```

```
1
8
[1, 2, 3, 4]
```

Exempel 2

För att räkna upp alla element i en lista kan vi göra på två sätt. Ett som påminner om det vi gjort innan
lista = [1, 2, 3, 4]
n = len(lista) #len(lista) ger antalet element i listan (listans längd)
for i in range(n):
print(lista[i])
samt ett där vi loopar igenom listan direkt.
for element in lista:
print(element)

Båda sätten har sina fördelar och nackdelar. Vilken som används beror på situationen.

```
1   lista = [1, 2, 3, 4]
2
3   n = len(lista)   #len(lista) ger antalet element i listan (listans längd)
4   for i in range(n):
5       print(lista[i])
6
7   for element in lista:
8       print(element)
```

```
1
2
3
4
1
2
3
4
```

Listfunktioner

För en lista vid namn v och ett element kallat element så finns bland annat följande funktioner.

v.append(element), lägger till element sist i listan
v.count(element), räknar förekomsten av element i listan
v.index(element), ger index för första förekomsten av element
len(v), ger listans längd
max(v), ger största elementet i listan
min(v), ger minsta elementet i listan
v.remove(element), tar bort första förekomsten av element
v.reverse(), vänder listan baklänges
v.sort(), sorterar listan i storleksordning
sum(v), ger summan av elementen i listan

Exempel 3
Nedan sker ett antal listoperationer. Försök förutse vad som skrivs ut.

```
tal_1 = [3, 7, 2, 9, 7]
tal_1.sort()
print(tal_1)
tal_2 = [] #Skapar en tom lista
for n in range(len(tal_1)):
tal_2.append(tal_1[n] * 2)
print(tal_2)
print(tal_1)
tal_1.reverse()
print(tal_1)
```

```
1   tal_1 = [3, 7, 2, 9, 7]
2   tal_1.sort()
3   print(tal_1)
4   tal_2 = []              #Skapar en tom lista
5   for n in range(len(tal_1)):
6       tal_2.append(tal_1[n] * 2)
7   print(tal_2)
8   print(tal_1)
9   tal_1.reverse()
10  print(tal_1)
```

```
[2, 3, 7, 7, 9]
[4, 6, 14, 14, 18]
[2, 3, 7, 7, 9]
[9, 7, 7, 3, 2]
```

Uppgift 1

Modifiera exempel 2 från träff 4 (finns på nästa bild) så att de inmatade talen istället sparas i en lista. Programmet ska även skriva ut största talet, minsta talet, medelvärdet och medianen.

Statistik i lista

```
talen = []
tal = int(input("Mata in ett positivt heltal: "))
while tal != 0:
    talen.append(tal)
    tal = int(input("Mata in ett positivt heltal: "))

#Sorterar talen i storleksordning
talen.sort()
#Tar reda på hur många tal som matats in
antal = len(talen)
#Summerar talen
summa = sum(talen)

if antal != 0:
    #Konrollerar om antalet tal är udda eller jämna
    if antal % 2 == 0:
        #Beräknar medianen om antalet tal är jämnt
        #Oservera att det går att utföra en beräkning av index i hakparentesen
        median = (talen[antal // 2 - 1] + talen[antal // 2]) / 2
    else:
        #Beräknar medianen om antalet tal är udda
        median = talen[antal // 2]
    print("Medelvärdet är", summa/antal)
    print("Medianen är", median)
    print("Största värdet är", max(talen))
    print("Minsta värdet är",min(talen))

print(talen)
```

```
Mata in ett positivt heltal: 5
Mata in ett positivt heltal: 9
Mata in ett positivt heltal: 7
Mata in ett positivt heltal: 3
Mata in ett positivt heltal: 0
Medelvärdet är 6.0
Medianen är 6.0
Största värdet är 9
Minsta värdet är 3
[3, 5, 7, 9]
```

Exempel 2 från träff 4

Programmet nedan beräknar medelvärdet av ett antal positiva heltal som matas in av användaren. Inmatningen avbryts när en nolla skrivs in.

antal = 0
summa = 0
tal = int(input(“Mata in ett positivt heltal: ”))
while tal != 0:
antal = antal + 1
summa = summa + tal
tal = int(input(“Mata in ett positivt heltal: ”))
if antal != 0:
print(“Medelvärdet är”, summa/antal)

Uppgift 2

Modifiera uppgift 4 c) från träff 4 så att primtalen sparas i en lista.

Primtal i lista

```
primtal = []
n = int(input("Antal primtal: "))

if n <= 0:
    print("Du vill inte se några primtal")
else:
    #Behandlar tvåan separat då det är det enda jämna primtalet
    primtal.append(2)
    tal = 3
    #Medan antalet primtal är mindre än n.
    while len(primtal) < n:
        arPrimtal = True
        #Vi behöver bara kontrollera delbarhet med de primtal vi har i listan
        for p in primtal:
            if tal % p == 0:
                arPrimtal = False
                #Avbryter testet (for-loopen) när vi hittat delbarhet
                break
        #Lägger till talet i listan om det är ett primtal
        if arPrimtal:
            primtal.append(tal)
        #Ökar talet med två då resten av primtalen är udda
        tal = tal + 2
    print("Primtalen är:")
    print(primtal)
```

```
Antal primtal: 89
Primtalen är:
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83
, 89, 97, 101, 103, 107, 109, 113, 127, 131, 137, 139, 149, 151, 157, 163, 167, 173, 17
9, 181, 191, 193, 197, 199, 211, 223, 227, 229, 233, 239, 241, 251, 257, 263, 269, 271,
 277, 281, 283, 293, 307, 311, 313, 317, 331, 337, 347, 349, 353, 359, 367, 373, 379, 3
83, 389, 397, 401, 409, 419, 421, 431, 433, 439, 443, 449, 457, 461]
```

Statistik: Python har en inbyggd statistikmodul. Vi får tillgång till den genom att importera den.

from statistics import *

Därefter kan vi använda funktionerna för medelvärde, median, typvärde och standardavvikelse för en lista.

mean(listnamn)
median(listnamn)
mode(listnamn)
stdev(listnamn)

Uppgift 3

statistikfunktionerna

Gör om uppgift 1, men använd nu statistikfunktionerna.

```
from statistics import *

talen = []
tal = int(input("Mata in ett positivt heltal: "))
while tal != 0:
    #Lägg till tal i listan
    talen.append(tal)
    tal = int(input("Mata in ett positivt heltal: "))

#Om listan inte är tom: len(talen) != 0
if len(talen) != 0:

    #Använder de fördefinierade funktionerna mean, median från statistics
    print("Medelvärdet är", mean(talen))
    print("Medianen är", median(talen))
    print("Största värdet är", max(talen))
    print("Minsta värdet är",min(talen))
```

```
Mata in ett positivt heltal: 3
Mata in ett positivt heltal: 4
Mata in ett positivt heltal: 7
Mata in ett positivt heltal: 9
Mata in ett positivt heltal: 0
Medelvärdet är 5.75
Medianen är 5.5
Största värdet är 9
Minsta värdet är 3
```

Exempel 4

Programmet nedan gör 1000 försök där varje försök undersöker hur många slag man behöver slå med en tärning innan man får en sexa.

from statistics import *
from random import randint
slagLista = [] #Skapar en tom lista att spara antalet slag i for i in range(1000):

tarning = 0 #Innan första slaget får värdet vara 0
antalSlag = 0 #Sätter antalet slag till 0
while tarning != 6:
tarning = randint(1,6)
antalSlag = antalSlag + 1
slagLista.append(antalSlag)

print(“Medel:”,mean(slagLista), “Standardavvikelse:”, stdev(slagLista))
print(“Median:”, median(slagLista), “Typvärde:”, mode(slagLista))

```
from statistics import *
from random import randint
slagLista = []        #Skapar en tom lista att spara antalet slag i
for i in range(1000):
    tarning = 0       #Innan första slaget får värdet vara 0
    antalSlag = 0     #Sätter antalet slag till 0
    while tarning != 6:
        tarning = randint(1,6)
        antalSlag = antalSlag + 1
    slagLista.append(antalSlag)

print("Medel:",mean(slagLista), "Standardavvikelse:", stdev(slagLista))
print("Median:", median(slagLista), "Typvärde:", mode(slagLista))
```

```
Medel: 6.108 Standardavvikelse: 5.535790908350945
Median: 4.0 Typvärde: 1
```

Omvandling till binära tal.

För att omvandla talet 13 till ett binärt tal (endast siffrorna 1 och 0 finns) kan man använda heltalskvoter och rester.

13 = 2*6 + 1 13 = 2*6 + 1 = 2*(2*3 + 0) + 1
6 = 2*3 + 0 ⇔ = 2*(2*(2*1 + 1) + 0) + 1
3 = 2*1 + 1 = 23
*1 + 22
*1 + 21
*0 + 20
*1
1 = 2*0 + 1

Alltså ger resterna det binära talet i omvänd ordning. Så 1310=11012

I ett program kan kan vi skriva 13 // 2 för att få heltalskvoten samt 13 % 2 för att få resten.

Uppgifter till nästa gång

4. Skriv ett program som omvandlar ett heltal skrivet på basen 10 till ett binärt tal genom att använda en lista. Det binära talet ska skrivas ut "fint", det vill säga som ett vanligt tal och inte som en lista, till exempel "100101". Vi vill påminna om att utskriftsfunktionen har ett frivilligt argument end=, som kan användas för att bestämma hur utskriften avslutas. Om argumentet utelämnas blir avslutet automatiskt en radbrytning. Exempelvis ger print("test", end=" ")ett mellanslag efter utskriften.

```
1   tal = int(input("Ange heltal att omvandla till binärt: "))
2
3   bas2 = []
4   if tal != 0:
5       bas2h = ''
6       while tal > 0:
7           #Bilda biten med den högsta bitvikten
8           bit = int(tal % 2)
9           #Dividera bort denna
10          tal = tal // 2
11          #Lägg in den beräknade biten i en lista
12          bas2.append(bit)
13      #Vänd på listan
14      #Biten med den högsta vikten lades ju in först
15      #Detta är inte nödvändigt om man väljer att skriva ut listan baklänges
        (nedan)
16      bas2.reverse()
17
18
19  print("Talet blir ", end="")
20  for bit in bas2:
21
22      #end="" gör så att raden inte bryts och att inget skrivs ut mellan tecknen.
23      print(bit, end="")
24  print(" i bas 2.")
25
26  #Om du inte vänder på listan:
27  #for i in range(len(bas2) - 1, -1, -1):
28      #print(bas2[i], end="")
```

```
Ange heltal att omvandla till binärt: 30
Talet blir 11110 i bas 2.
```

Uppgifter till nästa gång

5. a) Skapa en kortlek genom att använda en lista och skriv ut den blandad. I kortleken tar du hänsyn enbart till kortens valör. Du kan blanda en kortlek genom att importera funktionen shuffle från random. Den används genom instruktionen shuffle(lista).

```
from random import shuffle

kortlek = []

#Skapar en kortlek
for i in range(1,14):
    #Gör så att fyra kort av samma valör skapas
    for n in range(4):
        kortlek.append(i)

#Blandar kortleken
shuffle(kortlek)

#Skriver ut kortleken
print(kortlek)
```

```
[7, 3, 11, 5, 1, 2, 10, 10, 13, 8, 6, 9, 8, 8, 5, 2, 12, 4, 2, 12, 5, 6, 9, 11, 3, 1,
, 7, 12, 13, 4, 2, 10, 1, 13, 3, 13, 3, 7, 1, 5, 11, 4, 12, 9, 11, 7, 10, 6, 6, 8, 4]
```

b) Utvidga programmet så att du bestämmer sannolikheten för att de två första korten i kortleken bildar ett par.

```
from random import shuffle

kortlek = []

#Skapar kortleken
for i in range(1,14):
    for n in range(4):
        kortlek.append(i)

#Räknare för antalet par
par = 0

antalSimuleringar = 100000

#Simuelerar ett antal spel
for i in range(antalSimuleringar):
    #Blandar kortleken
    shuffle(kortlek)
    #Kontrollerar om de två första korten är ett par
    if kortlek[0] == kortlek[1]:
        par = par + 1

print("Sannolikheten att de två första korten är ett par är",
par/antalSimuleringar)
```

Lång tids väntan

```
Sannolikheten att de två första korten är ett par är 0.05798
```

c) Ändra programmet så att du istället bestämmer sannolikheten för att det bland de fem första korten i kortleken finns ett fyrtal.

```
from random import shuffle

kortlek = []

#Skapar kortleken
for i in range(1,14):
    for n in range(4):
        kortlek.append(i)

#Räknare för antalet fyrtal
fyrtal = 0
antalSimuleringar = 100000

#Simuleringen
for i in range(antalSimuleringar):
    #Blandar korleken
    shuffle(kortlek)

    hand = []
    #Väljer ut en hand på 5 kort
    for n in range(5):
        hand.append(kortlek[n])
    #Sorterar handen i storleksordning
    hand.sort()
    #Kontrollerar om vi har fyrtal
    #Det räcker att kontrollera likhet hand[0] == hand[3] eller hand[1] == hand
    [4]
    #Ex. [2, 2, 2, 2, 8] och [3, 7, 7, 7, 7]
    if hand[0] == hand[3] or hand[1] == hand[4]:
        fyrtal = fyrtal + 1

print("Sannolikheten att bland de fem första korten ha ett fyrtal är",
fyrtal/antalSimuleringar)
```

Lång tids väntan

```
Sannolikheten att bland de fem första korten ha ett fyrtal är 0.00021
```

d) Ändra programmet så att du istället bestämmer sannolikheten för att de fem första korten i kortleken bildar en stege.

```
from random import shuffle

kortlek = []

#Skapar kortleken
for i in range(1,14):
    for n in range(4):
        kortlek.append(i)

#Räknare för stege
stege = 0

antalSimuleringar = 10000000

for i in range(antalSimuleringar):
    #Blandar kortleken
    shuffle(kortlek)
    hand = []
    #Skapar en hand på 5 kort
    for n in range(5):
        hand.append(kortlek[n])
    #Sorterar korten i storleksordning
    hand.sort()
    #Kontrollerar om vi har stege
    if hand[0] + 4 == hand[1] + 3 == hand[2] + 2 == hand[3] + 1 == hand[4]:
        stege = stege + 1
    #Specialfallet med ess som 14
    elif hand[0] == 1 and hand[4] == 13 == hand[3] + 1 == hand [2] + 2 == hand
    [1] + 3:
        stege = stege + 1

print("Sannolikheten att bland de fem första korten ha en stege",
stege/antalSimuleringar)
```

Tog så lång tid att jag aldrig fick ut något svar, sänkte antalet simuleringar till 1000

```
Sannolikheten att bland de fem första korten ha en stege 0.007
```

Träff 6

Nedan finner du presentationen med genomgång, exempel och uppgifter. Du finner också ett uppgiftshäfte med uppgifter från olika kurser.

Mål med träff 6

Att kunna skiva pseudokod [6]

Att kunna använda programmering för att lösa matematiska problem

Att få förståelse för den typ av problem som den nya ämnesplanen ger möjlighet till [6]

Python

Undervisningstillfälle 6. Vad och varför?
Strategier för matematisk problemlösning inklusive modellering av olika situationer, såväl med som utan digitala verktyg och programmering.

Algoritmer

En algoritm är en ändlig lista med instruktioner som löser ett specifikt problem. Till exempel ett matlagningsrecept eller morgonrutinerna för en

gymnasieelev.

- Gå upp ur sängen
- Gå på toaletten
- Gå till köket
- Öppna kylskåpet
- Om det finns mjölk drick mjölk annars drick vatten
- Duscha så länge du inte är ren
- Tag på kläderna
- Cykla till skolan

Om att utveckla kod
För att kunna programmera en algoritm använder man ofta ett mellansteg kallat pseudokod för att strukturera sina tankar.

Identifiera problemet inklusive indata och utdata. Bryt ner problemet i delproblem. Bestäm i vilken ordning delproblemen ska lösas. Görs ofta med papper och penna.

Pseudokod - pq

Läs in a, b, c
Dela b med a
Dela c med a
Beräkna det som är under rottecknet. Om det under rottecknet är negativt skriv ut ”Reell lösning saknas” annars om det under rottecknet är noll skriv ut den enda lösningen annars beräkna lösningarna. skriv ut lösningarna

```
import math
a = float(input("a = "))
b = float(input("b = "))
c = float(input("c = "))
b = b / a
c = c / a
d = (b / 2)**2 – c
if d < 0:
 print("Reell lösning saknas")
elif d == 0:
 print("En dubbelrot: x =" , round( - b / 2 ,2))
else:
 x1 = round( -b / 2 +math.sqrt(d),2)
 x2 = round( - b / 2 -math.sqrt(d),2)
 print("Två lösningar: x1 =", x1 ," och x2 =", x2)
```

Kod - pq

Uppgift 1 - Skriv pseudokod

spelet risk

I spelet Risk tävlar två, eller flera, spelare om världsherraväldet. Vid ett moment i spelet ska en spelare anfalla medan den andra ska försvara. Anfallaren slår tre röda tärningar (1-6) och försvararen slår två blå tärningar

(1-6). Tärningarna sorteras sedan i fallande ordning. Därefter jämförs högst röd med högst blå, och näst högst röd med näst högst blå.

För att anfallaren ska vinna måste den röda tärningen visa fler prickar än den blå. I övriga fall vinner försvararen.

EXEMPEL Sorterat
Röd: 2, 6, 4 6, 4, 2
Blå: 4, 5 5, 4

I exemplet vinner röd det första paret (6 > 5) och blå det andra paret (4 <= 4). Spelomgången blev alltså oavgjord. Uppgift: Skriv ett program som uppskattar sannolikheten för följande händelser:

1. Röd vinner båda
2. Röd vinner en och blå vinner en
3. Blå vinner båda

Programmet ska fråga efter antal försök som ska genomföras.

```
from random import *

v_red = [0, 0, 0]
v_blu = [0, 0]
red_wins = 0
blu_wins = 0
draw = 0

n = int(input("Antal försök: "))

for i in range(n):
    #Slumpa fram tärningarnas antal ögon
    v_red[0] = randint(1, 6)
    v_red[1] = randint(1, 6)
    v_red[2] = randint(1, 6)
    v_blu[0] = randint(1, 6)
    v_blu[1] = randint(1, 6)
    #Sortera fälten med tärningarna
    v_red.sort()
    v_blu.sort()

    if v_red[2] > v_blu[1] and v_red[1] > v_blu[0]:
        red_wins = red_wins + 1     #Röd vinner
    elif v_blu[1] >= v_red[2] and v_blu[0] >= v_red[1]:
        blu_wins = blu_wins + 1     #Blå vinner
    else:
        draw = draw + 1             #Oavgjort

#Skriv ut sannolikheterna för de olika utfallen
print("Röd vinner båda: " + str(red_wins / n))
print("Blå vinner båda: " + str(blu_wins / n))
print("Oavgjort:        " + str(draw / n))
```

```
Antal försök: 8
Röd vinner båda: 0.375
Blå vinner båda: 0.25
Oavgjort:        0.375
```

Uppgifter till nästa gång

Välj några uppgifter från kompendiet som passar dig och lös dem.

Kompendiet innehåller problem som kan lösas med programmering, modellering

av olika situationer med hjälp av programmering och uppgifter där ett begrepp

utforskas med hjälp av programmering.

Uppgifterna är graderade med svårighetsgrad 1-3

Matematik grund

Tärningskast

Tärningskast 

Om du kastar två tärningar, vad är sannolikheten att summan är mer än sju? Varför får man olika svar när man kör programmet?

```
# importerar heltalsslumpare
from random import randint

# Matar in antalet simuleringar
n = int(input('Ange antal simuleringar: '))
# Sätter räknaren som
antal = 0

# En loop som upprepas n gånger
for i in range(n):
    # Slumpar fram första tärningen
    t1 = randint(1, 6)
    # Slumpar fram andra tärningen
    t2 = randint(1, 6)
    # Kontrollerar om summan är 7
    if t1 + t2 == 7:
        # Om summan är 7 räknas räknaren upp ett steg
        antal = antal + 1

# Skriver ut sannolikheten i procent
print('Sannolikheten för att tärningssumman är 7 är', round(antal/n*100, 2), '%')
```

```
Ange antal simuleringar: 100
Sannolikheten för att tärningssumman är 7 är 17.0 %
```

```
Ange antal simuleringar: 100
Sannolikheten för att tärningssumman är 7 är 20.0 %
```

```
Ange antal simuleringar: 1000
Sannolikheten för att tärningssumman är 7 är 17.2 %
```

```
Ange antal simuleringar: 1000
Sannolikheten för att tärningssumman är 7 är 15.3 %
```

Den teoretiska sannolikheten att summan är över sju är (5+4+3+2+1)/36=15/36=5/12=0,42

Den teoretiska sannolikheten att summan är sju är 6/36=1/6=0,17

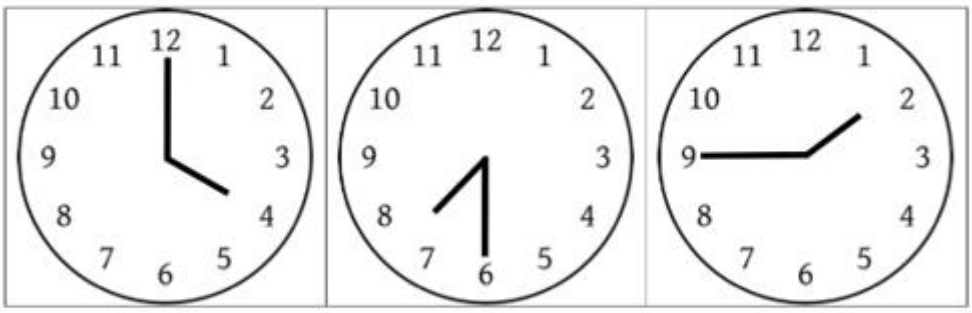
Vinkeln mellan visarna

Vinkeln mellan visarna

På en urtavla finns timvisare och minutvisare. Den korta är timvisaren och den långa är minutvisaren.

Uppgift 1

Nedan visas tre klockor. Beräkna vinkeln mellan timvisaren och minutvisaren för de tre klockorna.



Klockorna i figuren ovan visar klockslagen
4:00, 7:30 och 1:45.

Klockan kan alltså visa alla klockslag mellan 00:00 och 11:59.

Uppgift 2

Skriv ett program som tar emot timmar(0 – 11) och minuter (0 – 59) och som sedan beräknar vinkeln mellan timvisaren och minutvisaren.

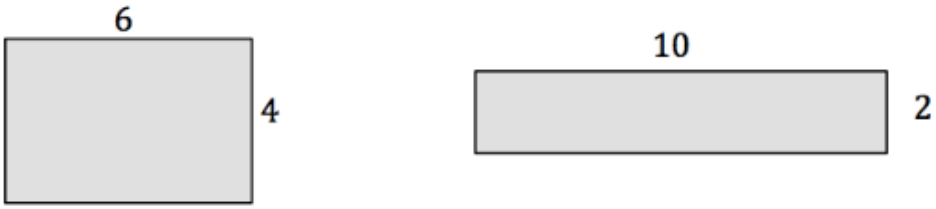
```
1   # Anger antal timmar
2   timmar = int(input('Ange antal timmar: '))
3   # Anger antal minuter
4   minuter = int(input('Ange antal minuter: '))
5
6   # Beräknar vinkeln på minutvisaren räknat från klockan 12
7   vMinut = minuter / 60 * 360
8   # Beräknar vinkeln på timvisaren räknat från klockan 12
9   vTimme = timmar / 12 * 360 + vMinut / 12
10
11  # Beräknar vinkeln mellan visarna
12  v = abs(vTimme - vMinut)
13
14  # Om vinkeln är större än 180 så har vi bräknat den större av de två möjliga vinklarna
15  # då beräknar vi den mindre
16  if v > 180:
17      v = 360 - v
18
19  print('Vinkeln mellan visarna är:', str(v) + '°')
```

```
Ange antal timmar: 5
Ange antal minuter: 20
Vinkeln mellan visarna är: 40.0°
>
```

Vänskapliga rektanglar

Vänskapliga rektanglar

Rektanglarna nedan är vänskapliga eftersom deras sidlängder är heltal och den vänstra rektangelns area är den högra rektangelns omkrets och den högra rektangelns area är den vänstra rektangelns omkrets.



Uppgift 1

Skriv ett program som skriver ut alla par av vänskapliga rektanglar med sidlängder mindre än 100.

Uppgift 2

Går det att hitta fler vänskapliga rektanglar än de som beräknas av programmet i uppgift 1.

```python
# Lista för de vänskapliga trianglarna
vanskapliga = []

# Räknar upp första sidan i första rektangeln
for a in range(1,101):
    # Räknar upp andra sidan i första rektangeln
    for b in range(a,101):
        # Beräknar area och omkrets i den första rektangeln
        area1 = a * b
        omkrets1 = 2 * (a + b)
        # Arean för andra rektangeln ska vara like med omkretsen för den första
        area2 = omkrets1
        # Lista för att kunna faktorisera arean för den andra rektageln för att få möjliga sidlängder
        faktor = []
        # Beräknar areans delare
        for n in range(1,area2+1):
            if area2 % n == 0:
                faktor.append(n)
        # Låter sida 1 i den andra rektangeln vara en av faktorerna
        for sida1 in faktor:
            # Beräknar sida 2 i den andra rektangeln
            sida2 = round(area2 / sida1)
            # Beräknar omkretsen i den andra rektangeln
            omkrets2 = 2 * (sida1 + sida2)
            # Kontrollerar om omkretsen för den andra rektangeln är lika stor som arean av den första
            if omkrets2 == area1:
                # Lägger till paret av vänskapliga rektanglar om så är fallet
                vanskapliga.append([[a,b],[sida1,sida2]])
                break

# Skriver ut de vänskapliga rektanglarna
for item in vanskapliga:
    print(item)
```

```
[[1, 34], [7, 10]]
[[1, 38], [6, 13]]
[[1, 54], [5, 22]]
[[2, 10], [4, 6]]
[[2, 13], [3, 10]]
[[3, 6], [3, 6]]
[[3, 10], [2, 13]]
[[4, 4], [4, 4]]
[[4, 6], [2, 10]]
[[5, 22], [1, 54]]
[[6, 13], [1, 38]]
[[7, 10], [1, 34]]
```

Aproximation av talet pi

Approximation av talet π

Det finns många sätt att uppskatta värdet på π. Ett sätt är att använda talföljder. Din uppgift är att skriva program för att beräkna ett närmevärde på π genom att använda en talföljd. De talföljder du ska använda är

Uppgift 1

Leibniz: $\pi/4 = 1 - 1/3 + 1/5 - 1/7 + \ldots$

Uppgift 2

Wallis: $\pi/2 = \frac{2\cdot2}{1\cdot3} \cdot \frac{4\cdot4}{3\cdot5} \cdot \frac{6\cdot6}{5\cdot7} \cdot \ldots$

Uppgift 3

Euler: $\pi^2/6 = 1 + 1/2^2 + 1/3^2 + 1/4^2 \ldots$

Uppgift 4

Viete: $2/\pi = \frac{\sqrt{2}}{2} \cdot \frac{\sqrt{2+\sqrt{2}}}{2} \cdot \frac{\sqrt{2+\sqrt{2+\sqrt{2}}}}{2} \cdot \ldots$

```
3.1416026536897204
3.1415769455087093
3.141583104230963
3.141592653589794
```

```python
#Närmevärden till pi
from math import *
#----------------------------------------
#a - Leibniz
pi = 0
f = 1

for i in range(1, 100000):
    pi = pi + f / (2 * i - 1)
    f = -f

print(4 * pi)
#----------------------------------------
#b - Wallis
pi = 1

for i in range(2, 100000, 2):
    pi = pi * i * i /((i - 1) * (i + 1))

print(2 * pi)
#----------------------------------------
#c - Euler
pi = 0

for i in range(1, 100000):
    pi = pi + 1 / (i * i)

print(sqrt(6 * pi))
#----------------------------------------
#d - Viete
pi = 1
num = sqrt(2)

for i in range(100000):
    pi = pi * num / 2
    num = sqrt(2 + num)

print(2 / pi)
```

Matematik 1

Basbyte

Basbyte

Uppgift 1

Skriv ett program som frågar om ett tal och talets bas (<10) och omvandlar talet till det decimala talsystemet.

Uppgift 2

Skriv ett program som frågar efter ett tal i bas 10 och en bas (<10) och omvandlar talet till den angivna basen.

Uppgift 3

Skriv ett program som frågar efter ett tal, talets bas (<10) och talets nya bas (<10) och omvandlar talet till den angivna basen.

```python
#Basbyte Uppgift 1

from math import *

tal = int(input("Tal? "))
bas = int(input("Bas? "))

bas10 = 0
i = 0
#Flagga som kontrollerar att inmatat tal kan vara skrivet i bas.
#Ej nödvändigt, men bra att känna till möjligheten.
flag = True

while tal > 0:
    #Bryt ut entalssiffran i tal genom att bilda resten
    bit = tal % 10
    #Flytta talet ett snäpp åt höger
    tal = tal // 10
    #Förbjudet! Om basen är n så är den högsta
    #bitvikten n - 1
    if bit >= bas:
        flag = False
        #Gå ur loopen
        break
    #Bygg upp det nya talet från höger till vänster
    bas10 = bas10 + bit * pow(bas, i)
    #Förbered samma procedur till nästa loopvarv
    i = i + 1

if flag:
    print(int(bas10))
else:
    print("Går ej")
```

```
Tal? 25
Bas? 2
Går ej
```

```
Tal? 10
Bas? 2
2
```

```
Tal? 556
Bas? 7
286
```

```python
print("Basbyte från en godtycklig bas(<10) till basen 10\n")

tal = input("Ange tal att omvandla: ")     #talet läses in som sträng
bas = int(input("Ange talets bas: "))
indata = True

siffror = []   #Tom lista
n= len(tal)
for i in range(n):
    siffror.append(int(tal[i]))    #siffrorna blir listelement
    if siffror[i] >= bas:
        print("Fel inmatning! Siffran", siffror[i], "finns inte i talsystemet
        med bas", bas)
        indata = False
        break

if indata:
    siffror.reverse()
    decimalt = 0
    for i in range(n):
        decimalt=decimalt+ siffror[i]*bas**i

    print("Talet", tal, "har värdet", decimalt, "i tiotalsystemmet")
```

```
Basbyte från en godtycklig bas(<10) till basen 10

Ange tal att omvandla: 23340
Ange talets bas: 5
Talet 23340 har värdet 1720 i tiotalsystemmet
```

```python
#Basbyte Uppgift 2
bas10 = int(input("Tal i bas 10: "))
nyBas = int(input("Ny bas: "))
nyTal = 0
i = 0

#Så länge det finns något kvar att omvandla
while bas10 > 0:
    #Bilda resten vid division med den nya basen
    bit = bas10 % nyBas
    #"Dividera bort", dvs bilda heltalkvoten då bas10 delas med nyBas
    bas10 = bas10 // nyBas
    #Bygg upp det nya talet från höger till vänster
    #Funktionen pow(a, b) ger a upphöjt till b
    nyTal = nyTal + bit * pow(10, i)
    #Ta ett "steg" till vänster
    i = i + 1

print(nyTal)
```

```
Tal i bas 10: 78
Ny bas: 2
1001110
```

```python
#Basbyte Uppgift 3
tal = int(input("Mata in talet: "))
frånBas = int(input("Mata in talets bas: "))
tillBas = int(input("Mata in ny bas: "))

#Programmet går via bas 10 och kombinerar alltså programmen från
#Uppgift 1 och Uppgift 2.

#För att göra en fin utskrift
sparaTal = tal

#Omvandla från frånBas till bas10
bas10 = 0
i = 0

while tal > 0:
    bit = tal % 10
    tal = tal // 10
    bas10 = bas10 + bit * pow(frånBas, i)
    i = i + 1

#Omvandla från bas10 till nyBas
nyttTal = 0
i = 0
while bas10 > 0:
    bit = bas10 % tillBas
    bas10 = bas10 // tillBas
    nyttTal = nyttTal + pow(10, i) * bit
    i = i + 1

print(sparaTal, "i bas", frånBas, "=", nyttTal, "i bas", tillBas)
```

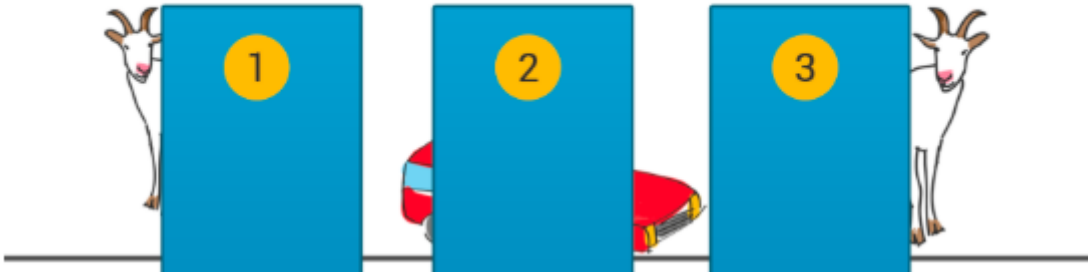
```
Mata in talet: 7820
Mata in talets bas: 10
Mata in ny bas: 4
7820 i bas 10 = 1322030 i bas 4
```

Monty Hall Problemet

Monty Hall Problemet

Problemet kan formuleras så här: I spelet får spelaren se tre dörrar - bakom en finns en bil, och bakom de två andra finns två getter. Spelet börjar med att spelaren väljer en dörr, utan att öppna den. Därefter öppnar spelledaren, som vet vad som finns bakom dörrarna, en av de två resterande dörrarna och visar att denna dörr inte innehåller vinsten. Spelledaren ger därefter spelaren ytterligare ett val, nämligen att byta dörr.
Är det då störst chans att vinna genom att byta dörr eller hålla kvar vid sitt ursprungliga val?

Skriv ett program som beräknar sannolikheten att vinna genom att byta dörr.




```python
from random import randint

# Nollställer antalet vinster vid byte
win = 0
# Antal simulationer
simulations = 10000

# Simuleringen
for i in range(simulations):
    # Nollställer dörrarna
    doors = ['Goat','Goat','Goat']
    # Slumpar fram vilken dörr bilen finns bakom
    car = randint(0,2)
    doors[car] = 'Car'

    # Väljer en dörr
    choice = randint(0,2)

    # Programledaren öppnar en dörr som innehåller en get och som inte är dörren som valdes
    # Programmet stegar sig genom dörrarna till en dörr som inte innehåller bilen eller är
    # alternativet som gästen valdes
    hostOpen = 0
    while doors[hostOpen] == 'Car' or hostOpen == choice:
        hostOpen = hostOpen + 1

    # Tar reda på vilken dörr som bytet ger
    # summan dörrnumrena för vald dörr och vilken värden öppnade ger vilken som man kan byta till
    Sum = choice + hostOpen
    if Sum == 1:
        change = 2
    elif Sum == 2:
        change = 1
    else:
        change = 0

    # Kontrollerar om det blir vinst vid dörrbyte
    if doors[change] == 'Car':
        win = win + 1

print("Vinstchans vid byte är", round(win / simulations * 100,1), "%")
```

```
Vinstchans vid byte är 66.7 %
```

Primtalsfaktorisering

Primtalsfaktorisering

Skriv ett program som tar ett positivt heltal och därefter skriver ut talets samtliga primtalsfaktorer i stigande ordning.

```python
#Primtalsfaktorisering
from math import *
x = int(input("Tal att primtalsfaktorisera: "))
v = []
max_faktor = int(sqrt(x))
sista_faktor = False

while not sista_faktor:

    #Flagga för hittad primtalsfaktor
    faktor_hittad = False

    i = 1

    steg = 1

    while not faktor_hittad and i <= max_faktor:
        i = i + steg
        #Fixar så att primtalsfaktorer > 2 endast är udda
        #När i når 3 ska vi fortsättningsvis bara kolla udda faktorer
        if i == 3:
            steg = 2

        #Om i är en faktor i x
        if x % i == 0:
            #så lägg till i som en faktor i listan
            v.append(i)
            #Dividera bort faktorn
            x = x // i
            #Beräkna ny högsta faktor för x
            max_faktor = int(sqrt(x))
            #Primtalsfaktor hittad
            faktor_hittad = True

        #Om max_faktor överskrids har vi hittat den sista primtalsfaktorn
        if i > max_faktor:
            v.append(x)
            sista_faktor = True
print(v)
```

```
Tal att primtalsfaktorisera: 36
[2, 2, 3, 3]
```

```python
# Inmatning av tal att faktorisera
tal = int(input("Ange positivt heltal att fakorisera: "))

# Skapar primtalslista
primtalsLista = [2]
# Sparar det största primtalet i listan
storstaPrimtal = 2
# Nästa tal efter 2
n = 3
# Upprepar så länge det största primtalet är mindre än eller lika med halva det inmatade talet
while storstaPrimtal <= tal//2:
    # Antar att talet är ett primtal
    arPrimtal = True
    # Loopar igenom primtalslistan
    for p in primtalsLista:
        # Kontrollerar om talet n delas av ett primtal
        if n % p == 0:
            # om det är delbart så är det inget primtal
            # och loopen avslutas
            arPrimtal = False
            break
    # Om n är ett primtal
    if arPrimtal:
        # Så läggs talet till i primtalslistan
        primtalsLista.append(n)
        # Det största primtalet blir nu det nya primtalet
        storstaPrimtal = n
    # Det nya talet att kontrollera blir nästa udda heltal
    n = n + 2

# Lista som sparar faktorerna
faktorer = []
# Loopar igenom primtalslistan
for p in primtalsLista:
    # Upprepar så länge primtalet delar talet
    while tal % p == 0:
        # Lägger till primtalet som faktor
        faktorer.append(p)
        # Dividerar bort primtalsfaktorn
        tal = tal / p
    # Kontrollerar om det finns fler faktorer att hitta
    if tal == 1:
        # Avbryter faktoriseringen om det inte finns fler faktorer
        break

# Skriver ut resultatet
if len(faktorer) == 0:
    print("Talet är ett primtal")
else:
    print("Talet primtalsfaktoriseras som",faktorer)
```

```
Ange positivt heltal att fakorisera: 36
Talet primtalsfaktoriseras som [2, 2, 3, 3]
```

Matematik 2

Ekvationssystem med två obekanta

Ekvationssystem med två obekanta

Skriv ett program som löser ett linjärt ekvationssystem på formen:

$$\begin{cases} a_1x + b_1y + c_1 = 0 \\ a_2x + b_2y + c_2 = 0 \end{cases}$$

Programmet ska skriva ut något av följande:

- Lösningen är (x, y)
- Lösning saknas
- Oändligt många lösningar

```python
#Ekvationssystem med två obekanta
a_1 = float(input("a1? "))
b_1 = float(input("b1? "))
c_1 = float(input("c1? "))
a_2 = float(input("a2? "))
b_2 = float(input("b2? "))
c_2 = float(input("c2? "))

if -a_1 / b_1 == -a_2 / b_2 and c_1 / b_1 == c_2 / b_2:
    print("Oändligt många lösningar")
elif -a_1 / b_1 == -a_2 / b_2:
    print("Lösning saknas")
else:
    x = (b_2 * c_1 / b_1 - c_2) / (-a_1 * b_2 / b_1 + a_2)
    y = -a_1 * x / b_1 - c_1 / b_1

    print("Lösning: (" + str(x) + ", " + str(y) + ")")
```

```
a1? 2
b1? 3
c1? 2
a2? 6
b2? 7
c2? 3
Lösning: (1.2500000000000004, -1.5000000000000002)
```

Felmarginal vid stickprovsundersökningar

Felmarginal vid stickprovsundersökningar

I ett land finns det *p* röstberättigade medborgare. I landet finns fem politiska partier. Medborgarna går till val vart fjärde år. Mellan valåren görs opinionsundersökningar med stickprov med storleken *n*.

Uppgift 1

För att undersöka hur fel det kan bli i när man gör en opinionsundersökning så kan man jämföra fördelningen i hela populationen och fördelningen i ett stickprov.

Utgå ifrån att det finns fem partier med följande fördelning i populationen

```
parti01 = 0.40
parti02 = 0.30
parti03 = 0.16
parti04 = 0.09
parti05 = 0.04

population = []

for i in range(int(parti01 * p)):
   population.append('p01')
for i in range(int(parti02 * p)):
   population.append('p02')
for i in range(int(parti03 * p)):
   population.append('p03')
for i in range(int(parti04 * p)):
   population.append('p04')
for i in range(int(parti05 * p)):
   population.append('p05')
```

Stickprovet kan nu fås genom att använda shuffle på populationslistan och välja ut de *n* första elementen.

Undersök hur differenserna mellan ett partis andel i stickprovet varierar.

Lämpliga mått på variansen kan vara differenserna standardavvikelse.

```
Populationens storlek? 1000
Stickprovets storlek? 100
För hela populationen:
Parti 1: 40.0%
Parti 2: 30.0%
Parti 3: 16.0%
Parti 4: 9.0%
Parti 5: 4.0%

För stickprovet:
Parti 1: 41.0%
Parti 2: 34.0%
Parti 3: 11.0%
Parti 4: 9.0%
Parti 5: 5.0%
```

```
#Felmarginal vid stickprov
from random import *
stickprov = []
populationsStorlek = int(input("Populationens storlek? "))
n = int(input("Stickprovets storlek? "))

# Partisympatiernas fördelning
parti01 = 0.40
parti02 = 0.30
parti03 = 0.16
parti04 = 0.09
parti05 = 0.04

# Lista för populationen
population = []

# Fyller på populationslistan med rätt antal partisympatisörer för varje parti
for i in range(int(parti01 * populationsStorlek)):
    population.append(1)

for i in range(int(parti02 * populationsStorlek)):
    population.append(2)

for i in range(int(parti03 * populationsStorlek)):
    population.append(3)

for i in range(int(parti04 * populationsStorlek)):
    population.append(4)

for i in range(int(parti05 * populationsStorlek)):
    population.append(5)

#Blanda populationen
shuffle(population)
#Ta ut stickprovet. v[a: b] returnerar en kopia av listan v från
#index a till b - 1.
stickprov = population[0: n]

print("För hela populationen:")
for i in range(1, 6):
    print("Parti " + str(i) + ": " + str(round(100 * population.count(i) / populationsStorlek, 1)) + "%")

print()

print("För stickprovet:")
for i in range(1,6):
    print("Parti " + str(i) + ": " + str(round(100 * stickprov.count(i) / n, 1)) + "%")
```

Jaktmodellering

Jaktmodellering

En förenklad modell för hur ett rovdjur jagar är att det alltid springer mot bytesdjuret, att bytesdjuret alltid springer i samma riktning och att både rovdjur och bytesdjur hela tiden springer med maximal hastighet.

Vi ska skriva ett program som simulerar en sådan jakt genom en stegmodell. En algoritm kan vara

Bytesdjurets startposition (x, y), hastighet (v)
Rovdjurets startposition (x, y), hastighet (v)
tiden = 0

medan rovdjuret inte är ikapp bytesdjuret
    Beräkna ny position för bytesdjuret
    Beräkna ny position för rovdjuret
    räkna upp tiden

Skriv ut hur lång tid det tog för bytesdjuret att komma ikapp.

```
# Importerar matematiska funktioner
from numpy import sign
from math import sqrt

# rovdjurets startkoordinater
x0Predator = 20
y0Predator = -100

# rovdjuret hastighet
vPredator = 18.8

# bytets startkoordinater
x0Prey = 0.0
y0Prey = 0.0

# bytets hastighet
vPrey = 13.3

# tidens steglängd
dT = 0.01

# starttid
t = 0.0

# Koordinatlistor för bytets och rovdjurets positioner
xPrey = [x0Prey]
yPrey = [y0Prey]
xPredator = [x0Predator]
yPredator = [y0Predator]

# räknare
i = 0

# startavstånd mellan bytet och rovdjuret
d = ((xPrey[i] - xPredator[i]) ** 2 + (yPrey[i] - yPredator[i]) ** 2) ** 0.5

# Ett godtyckligt villkor för när rovdjuret är tillräckligt nära för att anses ha fångat det
```

```
Speed of predator is 18.8 speed units. Speed of prey is 13.3 speed units.
The predator caught the prey after 9.36 time units.
The terminating distance was 0.00686 length units.
```

```
while d > (dT * vPredator / 10):
    # bytets förflyttning
    xPrey.append(xPrey[i] + vPrey * dT)
    yPrey.append(yPrey[i])

    # Kontroll om bytet och rovdjuret har samma x-koordinat dvs k-värdet är
    odefinierat
    if (xPrey[i] - xPredator[i]) == 0:
        # Rovdjurets förflyttning
        xPredator.append(xPredator[i])
        yPredator.append(yPredator[i] + sign(yPrey[i] - yPredator[i]) * vPredator *
        dT)
    else:
        # Rovdjurets förfluttning
        k = (yPrey[i] - yPredator[i])/(xPrey[i] - xPredator[i])
        x = xPredator[i] + sign(xPrey[i] - xPredator[i]) * sqrt((vPredator * dT) ** 2/
        (k ** 2 + 1))
        y = yPredator[i]+k*(x-xPredator[i])
        xPredator.append(x)
        yPredator.append(y)

    # Uppräkning av räknaren
    i = i + 1

    # avståndet mellan Preyn och räven
    d = ((xPrey[i] - xPredator[i]) ** 2 + (yPrey[i] - yPredator[i]) ** 2) ** 0.5

    # total förfluten tid
    t = t + dT

print("Speed of predator is " + str(vPredator) + " speed units. Speed of prey is " +
str(vPrey) + " speed units.")
print("The predator caught the prey after " + str(round(t, 3)) + " time units.")
print("The terminating distance was " + str(round(d, 6)) + " length units.")
```

```
#Jaktsimulering - likformighet
from math import *
#Inläsning av data för bytesdjuret (Prey = Byte)
x_prey = int(input("Bytesdjurets x-koordinat? "))
y_prey = int(input("Bytesdjurets y-koordinat? "))
v_prey = float(input("Bytesdjurets hastighet? "))
#Inläsning av data för rovdjuret (Predator = Rovdjur)
x_pred = int(input("Rovdjurets x-koordinat? "))
y_pred = int(input("Rovdjurets y-koordinat? "))
v_pred = float(input("Rovdjurets hastighet? "))

#Beräkna avståndet mellan djuren
dist = sqrt((x_pred - x_prey) ** 2 + (y_pred - y_prey) ** 2)
#Maxtid: 1h, tidssteg 0.1, starta med t = 0
dt = 0.1
t = 0

#Så länge avståndet mellan djuren är mer än 10% av v_prey och maxtid ej uppnådd
while dist > v_prey * dt and t < 3600:

    #Flytta fram bytesdjuret dx = vdt
    x_prey = x_prey + v_prey * dt

    #Beräkna avståndet mellan djurens punkter i x- resp. y-led
    delta_X = x_prey - x_pred
    delta_Y = y_prey - y_pred

    #Utnyttja likformighet för att bestämma rovdjurets förflyttningar
    #i x- resp. y-led
    #v_pred * dt = ds som är rovdjurets förflyttning längs linjen.
    #dx och dy är rovdjurets förflyttning i x- resp y-led på tiden dt.
    #Då gäller det att dx/ds förhåller sig som delta_X/dist.
    #Det gäller såklart också att dy/ds = delta_Y/dist.
    dx = delta_X * v_pred * dt / dist
    dy = delta_Y * v_pred * dt / dist

    #Flytta rovdjuret i x- och y-led
    #x_pred += dx är samma sak som x_pred = x_pred + dx vilket är ett
    #jättebra skrivsätt när variabelnamnen är långa
    x_pred += dx
    y_pred += dy

    #Beräkna nytt avstånd mellan djuren
    dist = sqrt((x_pred - x_prey) ** 2 + (y_pred - y_prey) ** 2)

    #Räkna upp tiden
    t = t + dt

if t >= 3600:
    print("Jakten misslyckades")
else:
    print("Det tog", int(t + 0.5), "sekunder")
```

```
Bytesdjurets x-koordinat? 20
Bytesdjurets y-koordinat? 20
Bytesdjurets hastighet? 35
Rovdjurets x-koordinat? 0
Rovdjurets y-koordinat? 0
Rovdjurets hastighet? 20
Jakten misslyckades
```

Matematik 3

Numerisk derivering

Numerisk derivering 

Utgå från följande närmevärden till $f'(x)$:

Metod 1

$$f'(x) \approx \frac{f(x+h) - f(x)}{h}$$

Metod 2

$$f'(x) \approx \frac{f(x) - f(x-h)}{h}$$

Metod 3

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h}$$

Uppgift 1

Skriv ett program som beräknar ett närmevärde till $f'(a)$ med de tre olika metoderna. Välj f så att du kan kontrollera resultatet av beräkningarna.

Testkör programmet för olika värden på h och jämför med det exakta värdet på derivatan $f'(a)$.

Uppgift 2

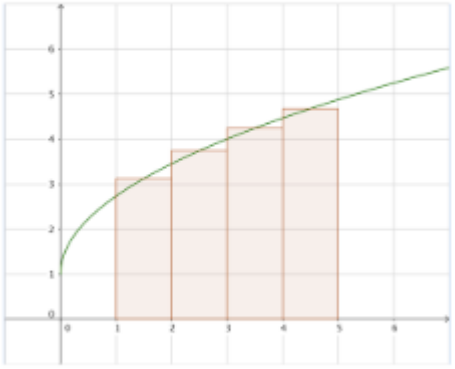
Komplettera programmet från uppgift 1 så att det skriver ut en tabell där närmevärdet till $f'(a)$ med de olika metoderna för h = 1, 0.1, ..., 0.000001.

Uppgift 3

Skriv ett program som bestämmer tangentens ekvation till kurvan för x = a.

Numerisk beräkning av integraler

Numerisk beräkning av integraler - Rektangelmetoden



Exempel

$$\int_1^5 (\sqrt{3x}+1)dx \approx f(1{,}5)\cdot 1 + f(2{,}5)\cdot 1 + f(3{,}5)\cdot 1 + f(4{,}5)\cdot 1$$

n = antal rektanglar

$$\Delta x = \frac{b-a}{n}$$

$$\int_a^b f(x)dx \approx \Delta x \cdot \sum_{i=0}^{n-1}\left(f\left(a + i\cdot\Delta x + \frac{\Delta x}{2}\right)\right)$$

Uppgift

Skriv ett program som beräknar ett närmevärde till en integral.
Programmet ska fråga efter integrationsgränserna och antal rektanglar.

Testkör programmet och jämför resultatet med det exakta värdet på en känd integral.
Hur många rektanglar tycker du behövs för att metoden ska ge ett godtagbart värde på integralen?

Approximation av talet e

Approximation av talet e

Derivatan av exponentialfunktionen $f(x) = a^x$ är $f'(x) = k \cdot a^x$. Konstanten e är definierad som det tal som gör att $k = 1$.

Uppgift 1

Skriv ett program där användaren gör upprepade gissningar på talet a och som därefter skriver ut f(x) och f'(x). f'(x) beräknas med symmetrisk differenskvot.

För vilket värde på a blir f(x)=f'(x).

Uppgift 2

Inspirerad av Skolverkets workshop om programmering i matematikkurser 2017-09-21

Skriv ett program som bestämmer ett närmevärde på e.

Ledning: Börja med att gissa ett värde på e. Undersök derivatans värde (med hjälp av lämplig förändringskvot) i en punkt. Öka/minska sedan värdet på e tills derivata och funktionens värde överensstämmer tillräckligt bra. Tänk på att det går bra att undersöka överensstämmelsen för ETT värde på x.

```python
#Approximation av talet e
from math import *
#Programmet låter användaren gissa sig fram till ett tal a som uppfyller
#det i uppgiften givna villkoret.

#fun - funktionsvärdet och der - derivatans värde beräknad med
#symmetrisk differenskvot med den fördefinierade funktionen pow(a, b)
#Startvärdena kan väljas hur som helst bara de inte sätts lika.
fun = 0
der = 1

#Räkna antal gissningar
n = 0

#Programmet håller på till den tredje decimalen är korrekt!
while abs(fun - der) > 0.001:
    #Användaren gissar
    a = float(input("Skriv in gissning ( > 0): "))
    n += 1
    #Beräkna funktionens värde och derivatan med ändringskvot.
    h = 0.0001
    fun = pow(a, 1)
    der = (pow(a, 1 + h) - pow(a, 1 - h)) / (2 * h)
    print("Derivatan: " + "%.3f" % der)

    if abs(fun - der) > 0.001:
        print("Bättre kan du!")

print("Du gissade", n, "gånger")


#-------------------------------------------------------

#Man kan låta en elevgrupp testköra programmet och därefter
#visa grafiskt hur gissningen gick.
#Programmet kan enkelt modifieras till att beräkna fun / der och
#istället söka det värde på a som ger kvoten 1
```

```
Skriv in gissning ( > 0): 3
Derivatan: 3.296
Bättre kan du!
Skriv in gissning ( > 0): 2
Derivatan: 1.386
Bättre kan du!
Skriv in gissning ( > 0): 2.5
Derivatan: 2.291
Bättre kan du!
Skriv in gissning ( > 0): 2.78
Derivatan: 2.842
Bättre kan du!
Skriv in gissning ( > 0): 2.71828
Derivatan: 2.718
Du gissade 5 gånger
```

```python
# funktion som beräknar symmetrisk ändringskvot
def andringskvot(a):
    return (a ** (x + h) - a ** (x - h))/(2 * h)


# Väljer ett x-värde att beräkna e med
x = 1

# Väljer hur många decimaler på e som ska beräknas
noggranhet = int(input('Ange antal decimaler på e: '))
# Gissar ett värde på e
e = float(input('Gissa ett värde på talet e: '))

# Sätter ett löjligt stort värde på h baserat på noggrangeten
h = 10 ** -(noggranhet * 2)

# Loop som upprepas medan e:s ändringskvot för det givna x-värdet inte får samma avrundning som e^x
while round(andringskvot(e), noggranhet+1) != round(e ** x, noggranhet + 1):
    # Beräknar hur värdet på ska förändras (intevallhalvering)
    steg = abs(andringskvot(e) - e ** x) / 2
    # Ändrar värdet på e beroende på om ändringskvoten eller e^x är störst
    if andringskvot(e) > e ** x:
        e = e - steg
    else:
        e = e + steg
    # Skriver ut värdet på e
    print(round(e, noggranhet))
```

```
Ange antal decimaler på e: 5
Gissa ett värde på talet e: 3
2.85208
2.78356
2.75053
2.73431
2.72627
2.72227
2.72028
2.71928
2.71878
2.71853
2.71841
2.71834
2.71831
2.7183
2.71829
2.71828
2.71828
```

Matematik 4

Komplexa tal på olika former

Matematik 4

Komplexa tal på olika former

Uppgift 1

Skriv ett program som omvandlar ett komplext tal på rektangulär form till polär form. Realdelen och imaginärdelen till det komplexa talet skall läsas in och därefter ska talet skrivas ut på polär form.

Lös uppgiften så att en utskrift görs där argumentet anges i radianer och en utskrift där argumentet anges i grader. Se till att båda utskrifter blir med lämpligt antal decimaler.

Uppgift 2

Skriv ett program som omvandlar ett komplext tal på polär form till rektangulär form. Programmet ska fråga användaren om argumentet ska anges i radianer eller i grader och sedan läsa in vektorns längd och argumentet.

Därefter ska det komplexa talet skrivas ut på rektangulär form.

```
#Komplexa tal på olika former
from math import *

# a
a = float(input("Rez? "))
b = float(input("Imz? "))

r = round(sqrt(a ** 2 + b ** 2), 2)

#atan2(b, a) ger principalargumentet -pi <= v <= pi
v = atan2(b, a)

#Om v < 0 adderas 2pi för att få argumentet 0 <= v < 2pi
if v < 0:
    v = v + 2 * pi

argrad = str(round(v, 2))

#u"\u00B0" är gradertecknet
argdeg = str(round(degrees(v), 2)) + u"\u00B0"

#Fixa till strängarna som ska skrivas ut
#Rektangulär, polär och potensform enligt Euler
rek  = str(a) + " + " + str(b) + "i"
pol1 = str(r) + "(cos(" + argrad + ") + isin(" + argrad + "))"
pol2 = str(r) + "(cos(" + argdeg + ") + isin(" + argdeg + "))"
pot  = "e^(i" + argrad + ")"

print(rek + " = " + pol1 + " = " + pol2 + " = " + pot)
print()

```

```
Rez? 5
Imz? 3
5.0 + 3.0i = 5.83(cos(0.54) + isin(0.54)) = 5.83(cos(30.96°) + isin(30.96°)) = e^(i0.54)
```

```
# b
r = float(input("Absolutbelopp? "))
typ = input("Argument i rad eller deg <rad/deg>? ")
v = float(input("argz? "))

#Vinklar representeras i interna funktioner alltid i radianer.
#För rad -> deg används degrees(v i rad)
#För deg -> rad används radians(v i deg)

if typ == "deg":
    v = radians(v)

a = round(r * cos(v), 2)
b = round(r * sin(v), 2)

print("z = " + str(a) + " + " + str(b) + "i")
```

```
Absolutbelopp? 7
Argument i rad eller deg <rad/deg>? deg
argz? 34
z = 5.8 + 3.91i
```

Längden på en hängande kedja

Längden på en hängande kedja

från Skolverkets workshop om programmering i matematikkurser 2017-09-21

En fritt hängande kedja har en höjd y över marken som kan beskrivas med funktionen

$$y(x) = \frac{a}{2}(e^{x/a} + e^{-x/a}) + h$$

där x är avståndet i sidled från kedjans lägsta punkt. Anta att avståndet mellan stolparna är $2,00$ m, att den lägsta punkten är mitt emellan stolparna, och att värdena $a = 1,30$ m och $h = -0,70$ m. Hur lång är då kedjan? Svara med två decimalers noggrannhet.

En lösning är att dela in kedjan i små intervall och beräkna längden på varje intervall som om det är en rät linje.

```
#Längden på en kedja
#Generaliserad till längden på en kurva
from math import *

def f(x):
    # y = f(x)
    # Funktionen returnerar y för x.
    # Om du vill ha en annan funktion
    # skriver du in den istället.

    y = 0.65 * (exp(x / 1.3) + exp(-x / 1.3)) - 0.7
    return y


a = float(input("Undre gräns: "))
b = float(input("Övre gräns : "))
n = int(input("Antal delintervall: "))

dx = (b - a) / n
l = 0
x = a

for i in range(n):
    #Använd avståndsformeln och beräkna länden av kurvelementet dl
    dl = sqrt(dx ** 2 + (f(x + dx) - f(x)) ** 2)
    #Öka l med dl
    l = l + dl
    #Öka x med dx
    x = x + dx

print("Kurvans längd:", l)

#För att få längden på kurvan y = f(x) beräknar du
#integralen a till b sqrt(1 + (f'(x))^2)dx.
#Det görs förstås enklast numeriskt på räknaren
#eller med GeoGebra. Eller med programmet "Rektangel-
#metoden" (Se lösningsförslag.).
```

```
Undre gräns: 3
Ovre gräns : 40
Antal delintervall: 3
Kurvans längd: 14990650956305.135
```

Polynomdivision

Polynomdivision

Uppgift 1

Fundera hur algoritmen för polynomdivision fungerar. Gör några exempel. Hur får du fram kvot och rest i varje steg?

Uppgift 2

Skriv ett program som utför en polynomdivision genom att spara polynomets koefficienter i en lista.

```
# Polynomdivision
# Helt onödig funktion som skriver ut polynom snyggt
def prettyprint(lista):
    for j in range(len(lista)):
        if lista[j] != 0:
            if lista[j] < 0 and j != 0:
                print(' - ', end='')
                if lista[j] != -1:
                    print(-lista[j], end='')
                elif j == len(lista) - 1:
                    print(-lista[j], end='')
            elif lista[j] > 0 and j != 0:
                print(' + ', end='')
                if lista[j] != 1:
                    print(lista[j], end='')
                elif j == len(lista) - 1:
                    print(lista[j], end='')
            elif lista[j] != 1 and lista[j] != -1:
                print(lista[j], end='')
            if len(lista)-1-j > 1:
                print('x^', end='')
                print(len(lista)-1-j, end='')
            elif len(lista)-1-j == 1:
                print('x', end='')


# Nollställer nämnare och täljare
numerator = []
denominator = []

# Matar in täljarens polynpm
nGrad = int(input('Ange täljarens gradtal: '))
for n in range(nGrad + 1):
    print('Ange koefficient för x^', end='')
    numerator.append(float(input(str(nGrad - n) + ': ')))

print()
```

```
Ange täljarens gradtal: 3
Ange koefficient för x^3: 4
Ange koefficient för x^2: 2
Ange koefficient för x^1: 2
Ange koefficient för x^0: 2

Ange nämnarens gradtal: 1
Ange koefficient för x^1: 4
Ange koefficient för x^0: 1

(4.0x^3 + 2.0x^2 + 2.0x + 2.0)/(4.0x + 1.0)=

x^2 + 0.25x + 0.4375 + (1.5625)/(4.0x + 1.0)
```

Forts

```
39  # Matar in nämnarens polynom
40  tGrad = int(input('Ange nämnarens gradtal: '))
41  for n in range(tGrad + 1):
42      print('Ange koefficient för x^', end='')
43      denominator.append(float(input(str(tGrad - n) + ': ')))
44
45  # Beräknar hur många gånger divisionsalgoritmen ska upprepas
46  upprepning = len(numerator) - len(denominator) + 1
47
48  # Nollställer kvot och rest
49  kvot = []
50  rest = []
51
52  # Sätter resten lika med täljaren
53  for i in range(len(numerator)):
54      rest.append(numerator[i])
55
56  # Polynomdivisionen
57  for i in range(upprepning):
58      # Nollställer mellansteget
59      mStep = []
60      #Beräknar kvoten
61      kvot.append(rest[0]/denominator[0])
62      # Multiplicerar kvoten med nämnaren och sparar det som ett mellansteg
63      for n in range(len(denominator)):
64          mStep.append(kvot[i] * denominator[n])
65      # Drar bort mellansteget från resten
66      for n in range(len(mStep)):
67          rest[n] = rest[n] - mStep[n]
68      # Raderar positionen i resten som nu blivit noll.
69      del rest[0]
70
71  # I vissa fall finns det en rest där en nolla finns på en position innan den
    "verkliga" resten
72  # Dessa nollor raderas i detta steg
73  while len(rest) > 1 and rest[0] == 0:
74      del rest[0]
75
76  # Skriver ut kvot och rest
77  print()
78  print('(', end='')
79  prettyprint(numerator)
80  print(')/(', end='')
81  prettyprint(denominator)
82  print(')= ', end='\n \n')
83
84  prettyprint(kvot)
85  if rest[0] != 0:
86      print(' + (', end='')
87      prettyprint(rest)
88      print(')/(', end='')
89      prettyprint(denominator)
90      print(')')
```

```
Ange täljarens gradtal: 3
Ange koefficient för x^3: 4
Ange koefficient för x^2: 2
Ange koefficient för x^1: 2
Ange koefficient för x^0: 2

Ange nämnarens gradtal: 1
Ange koefficient för x^1: 4
Ange koefficient för x^0: 1

(4.0x^3 + 2.0x^2 + 2.0x + 2.0)/(4.0x + 1.0)=

x^2 + 0.25x + 0.4375 + (1.5625)/(4.0x + 1.0)
```

Matematik 5

Kast med fyra tärningar

Kast med fyra tärningar

Skriv program som löser uppgifterna.

a) Hur stor är sannolikheten för att exakt tre tärningar visar samma antal ögon?

b) Hur stor är sannolikheten att det blir två par (ej fyra lika).

c) Hur stor är sannolikheten att tärningarnas antal ögon är konsekutiva (direkt på på varandra följande)?

d) Hur stor är sannolikheten att samtliga fyra tärningar visar olika antal ögon?

Eulers stegmetod

Eulers stegmetod

Uppgift 1

Skriv ett program som använder Eulers stegmetod för att lösa en godtycklig differential-ekvation av första graden.

Eulers stegmetod: $y(x+\Delta x) = y(x) + y'(x)\Delta x$

Programmet ska fråga efter

x_0 Startvärde för x
y_0 Startvärde för y
x_{slut} Slutvärde för x
Δx Steglängd

Tips

Testa programmet för en differentialekvation som enkelt kan lösas algebraiskt.

Uppgift 2

Spara x- och y-värdena i en lista och plotta lösningskurvan för $x_0 \leq x \leq x_{slut}$.

Fibonaccis talföljd

Fibonaccis talföljd 

Uppgift 1

Skriv ett program som genererar de *25* första talen i Fibonaccis talföljd:

[1, 1, 2, 3, 5, 8, 13, ...]

Uppgift 2

Undersök kvoten a_{n+1} / a_n

Vad händer? Vilket är talet?

Uppgift 3

Plotta a_n mot n.

Sannolikheter vid kortspel

Sannolikheter vid kortspel 

Programmering är ett lämpligt sätt att simulera fram sannolikheter för pokerhänder.

Uppgift 1

Skriv ett program som konstruerar en kortlek där varje kort har både en färg och en valör.

Uppgift 2

Använd din "kortlek" från uppgift 1 och skriv ett program som beräknar sannolikheten för att få

a) färg

b) stege

c) både hjärter ess och spader ess

vid slumpvis dragning av 5 kort i en kortlek.

Uppgift 3

Skriv en funktion som tar en lista med fem spelkort (uppgift 1) och som ger returvärden 1 om det finns ett triss och annars 0.

Uppgift 4

Skriv ett program för att simulera ett spel (med en kortlek) mellan 3 personer. Hur vanligt är det att exakt 2 personer har triss?

Andra ordningens differentialekvationer

Andra ordningens differentialekvationer

Eulers stegmetod för lösning av första ordningens differentialekvationer kan utvidgas till differentialekvationer av andra ordningen. Metoden är liknande. Man löser ut andraderivatan, inför en ny variabel för förstaderivatan. Sedan stegar man upp både funktionen och förstaderivatan.

Här tittar vi på en vikt som hänger i en fjäder och oscillerar. Enligt Hookes lag blir fjäderkraften $F = -k \cdot x$ den är också den enda kraften som verkar i en första approximation, kombinerat med Newtons 2:a lag, $F_R = m \cdot a$, får vi

$$m \cdot a = -k \cdot x$$
$$m \cdot x'' = -k \cdot x$$

alltså

$$x'' = v' = -(k/m) \cdot x$$
$$x' = v$$

I ögonblicket fjädern släpps är fjädern utsträckt med sträckan a från jämviktsläget och hastigheten är noll. Vi får därför följande stegmodell.

$$x_{n+1} = x_n + x' \cdot \Delta t \text{ med } x_0 = a$$
$$v_{n+1} = v_n - v' \cdot \Delta t \text{ med } v_0 = 0$$

Skriv ett program som sparar x, v och t -värdena i tre listor för tiden $t = 0$ s till $t = 2$ s. Rita upp dem. Studera även hur valet av Δt påverkar lösningen. Sätt $k = 10$ N/m, $m = 0.1$ kg och $a = 10$ cm.

För plottning kan följande paket importeras:

```
import matplotlib.pyplot as plt
```

För att få utskrift i koden skrivs:

```
plt.scatter(x-värden,y-värden,s=1)  #s är storleken på pricken
plt.show()
```